NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

HIERARCHICAL IMAGE SEGMENTATION
TO
INFARED IMAGES

by

James D. Bloomquist

June 1985

Thesis Advisor: C. H. Lee

Approved for public release; distribution is unlimited

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)*: Hierarchical Image Segmentation to Infared Images	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis, June, 1985
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): James D. Bloomquist	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943-5100	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943-5100	12. REPORT DATE: June, 1985
	13. NUMBER OF PAGES: 70
14. MONITORING AGENCY NAME & ADDRESS *(If different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*: Unclassified
	15a. DECLASSIFICATION/ DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*
Approved for public release; distribution is unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*
Image segmentation, hierarchiacal, infared image, Quad Split scope view, recursive segmentation

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Recursive image segmentation with hierarchical scope views is a new technique in image processing which systematically divides an image into smaller and smaller quadrants with each region within the quadrants having a structured descriptor. The regions are then processed to remove noise and to check for connected regions across the quadrant boundaries. The quadrants are then brought back together with the obsecured target

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

enhanced and distinguishable from the background. Infared image data of five different ship targets with the associated noise and interference was processed by this technique with the target information being greatly enhanced. The procedures developed to evaluate the data were found to be inadequate for the task, neccessitating hand evaluation to extract the target. The technique was proven to be a viable solution for extracting target information from infared data, but very slow processing times and inadequate evaluation procedures limit the usefulness of the program in its present form.

Approved for public release; distribution is unlimited.

Hierarchical Image Segmentation to Infared Images

by

James D. Bloomquist
Lieutenant, United States Navy
B.S., LaVerne College, 1976

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN EIECTRICAL ENGINEERING

from the

NAVAI POSTGRADUATE SCHOOL
June 1985

ABSTRACT

Recursive image segmentation with hierarchical scope views is a new technique in image processing which systematically divides an image into smaller and smaller quadrants with each region within the quadrants having a structured descriptor. The regions are then processed to remove noise and to check for connected regions across the quadrant boundaries. The quadrants are then brought back together with the obscured target enhanced and distinguishable from the background. Infared image data of five different ship targets with the associated noise and inteference was processed by this technique with the target information being greatly enhanced. The procedures developed to evaluate the data were found to be inadequate for the task, neccessitating hand evaluation to extract the target. The technique was proven to be a viable solution for extracting target information from infared data, but very slow processing times and inadequate evaluation procedures limit the usefulness of the program in its present form.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

Many different approaches exist in the field of image processing to extract meaningful information from such sources as aerial phctographs, infared images, or TV camera images. The major procedures of a typical image processing system are digitization, preprocessing, segmentation, feature extraction, and understanding. As presented in [Ref. 1], the purpose of segmentation is to partition the image into meaningful regions which depend on the prcblem being considered. In aerial reconnaissance, for example, a practical application of image segmentation may be to extract regions corresponding to terrain, industrial sites, or ships of interest.

There are two catagories of approaches for the segmentation process. The first category deals with methods which are based on examinirg an image on a point-by-point basis, called a point dependent process. An example of a point dependent process would be gray scale thresholding. The second category, referred to as a region dependent process, deals with techniques which utilize the image formation in a prescribed neighborhcod. Examples of this category of approach are based on edge detection, boundary finding, or region growing which attempt to group points with similar characteristics into regions.

A new method of segmentation utilizing methods of both catagories is by gray scale thresholding at hierarchical scope views [Ref. 2], which breaks an image into small quadrants with the regions within the quadrants having a structured descriptor and region number. This method uses many of the techniques first proposed by Ohlander (see [Ref. 3]) and then formulated into the MOOSE program by Shafer [Ref. 3].

Additional segmentation procedures are then applied to these small regions to pull out details and associate regions on oppsite sides of the boundaries. The segmented pieces are then processed to ascertain whether the region is valid or noise. All the pieces are then brought back together with the meaningful objects more prominently displayed.

Previous segmentation processes have not provided good results when using infared data as the input. Interference caused by such variables as temperature differences, clouds, and target variations have caused the segmentation processes to fail in one way or another. A new program, called QUAD SPLIT, will be tested with infared data input of ships in an attempt to produce meaningful and useful information. To extract the information, new procedures will be required to take the output from the QUAD SPLIT program and process it into a form that can be readily understood. The formulation and testing of these new procedures, and the testing of the QUAD SPLIT program utilizing infared data of ships as the input is the basis for this research work.

An overview of the recursive image segmentation methods utilized and their limitations are given in Chapter II, while a more detailed treatment of recursive image segmentation at hierarchical scope views is presented in Chapter III. The problems associated with the program QUAD SPLIT and the development of new procedures to remedy these problems are the subjects of Chapter IV. The testing of the revised program with infared image data and the evaluation of the processed data is covered in the final chapter.

The evaluation of the processed infared data confirmed that recursive image segmentation at hierarchical scope views is a viable technique for extracting target information from noisy infared images. The procedures developed for the evaluation proved to be inadequate which required

that visual and hand calculations be conducted to effectively extract the target. These inadequacies could be easily remedied since the required information is present in raw form and only requires proper processing. A major concern noted during testing is the processing time required for each image. For complicated images with numerous quadrant segmentations and boundary checks, processing time can be in excess of twenty minutes.

II. RECURSIVE IMAGE SEGMENTATION METHODS

This chapter is provided to give a brief overview of two of the methods used in this work for the segmentation of image data. The first section is an introduction to the MOOSE program and some of its limitations. The second section then discusses a method by which two of the MOOSE limitations can be remedied.

A. RECURSIVE SPLITTING

1. Assumptions

MOOSE is a program which implements a segmentation procedure using the assumptions proposed by Ohlander. The primary assumption for this technique is that the surfaces in the image will be represented by connected sets of pixels with a very close gray scale measure. The data used throughout this work are infared images represented as a 0 to 256 gray scale over a plane. The following additional assumptions are related to overcoming some of the flaws in the primary assumption: (1) Each of the connected sets of pixels, forming a patch, is assumed to produce a gaussian peak when histogrammed over the gray scale; (2) The histogram of a collection of these patches is assumed to clearly indicate a separation between peaks, with the relative minima of the histogram lying between these peaks rather than cutting peaks in the middle.

2. Algorithm

The basic idea of the algorithm is to split the image into regions, then split these regions into smaller regions, and so on, until only very small regions remain. The structure is thus recursive.

An image is considered as consisting of only one region at the beginning. If the region area is larger than a constant minimum size criterion, it will be segmented further. A histogram of the region is then calculated and the shape of the histogram is analyzed to yield a set of peak intervals. The relative minima between the peaks is then used to produce good thresholds levels. The region is then divided by these threshold levels into candidate patches that are each assigned a distinct numeric score. The patches are then collected by a connecting region procedure to reveal their geometric relationship. A patch must have an area greater than some constant minimum noise area, otherwise it is considered to be noise and merged with the surrounding region. The patches are then stored in a region record. This is then repeated for other regions until no further regions require splitting. the procedure and associated criteria are listed as follows:

Procedure	Criteria
Fetch a region	Minimum area
Histogramming	
Peak selection	
Thresholding	
Connected patch finding	
Noise elimination	Minimum area
Region collection	
Repeat until region can not be split further.	

3. Limitations

This recursive method has been applied yielding better results than the simple thresholding of gray scale approach. But there remain four main limitations:

a. Majority Rule Problem

The majority rule problem occurs when large and small objects (or areas) occur in the same region. Large objects will show up strongly in the histogram and will dominate. The smaller objects may not cause strong peaks in the histogram and thus may be covered by the large distributional spread of the larger objects. Therefore, these smaller objects will not appear in the segmented output.

b. No Well Defined Peaks

If there are no well defined peaks in the histogram then this procedure is ineffective. An image full of many small objects (i.e., cars in a parking lot) has a histogram which tends to have a relatively broad and flat distribution with no dominate peaks or valleys. Therefore these objects will be lost in the output.

c. Region Combination

Regions selected in early stages cannot be combined in later stages. The region boundary formed in the early stages will never change later on. Only more new boundaries will show up inside the old boundary.

d. Constant Gradient

Regions with constant intensity gradient cannot be detected. This can result in the splitting of an object that has a constant gray scale gradient instead of recognizing it as one object.

B. RECURSIVE SPLITTING AT HIERARCHICAL SCOPE VIEWS

1. Generalities

This approach was developed and implemented to avcid the first two disadvantages of MOOSE. The MOOSE algorithm is inccrporated into the program with additional tests and procedures added. It uses a hierarchy of scope views with a large scope view (large area) at high levels and small scope views (small areas) at lower levels. A quad tree (4 branch) development is used for splitting. Starting with a 256 X 256 pixel representation, the image is divided into 4 quadrant scope views by recursive image splitting. A test is then performed on the quadrant scope view to determine if further splitting is required. If the test results are acceptable, the quad tree development for this quadrant scope view stops at this point and is called a terminating node. If the test is not satisfactory for the scope view, then it is further divided into four more quadrant scope views. Each of these is tested in turn which will either terminate or again be split into four quadrant scope views depending on the test result. This continues until a minimum area criteria is violated for splitting or all scope views test satisfactorily and terminate.

The test used to check for acceptable results is based on the spread width of the histogram generated for the scope view. An object containing a large number of objects and a wide variation of gray scale levels would tend to give a broad distribution histogram with no well defined peaks. If the spread width exceeds a maximum value set up by the test, then the test is considered not satisfactory and the scope view will be split into quadrant scope views. The test is then performed on these smaller scope views. Thus because the larger quadrants are broken up into smaller quadrants with a better possibility of finding well defined

histogram peaks, the majority rule and no well defined peaks problems associated with MOOSE are relaxed.

2. Boundary Problems

One of the major problems of quad tree generation is that discontinuous boundaries may exist from one scope view to another. Two neighboring scope views with adjacent borders sometimes have different peak interval values which may lead to a non-closed region boundary in one scope view being discontinuous across the border. A boundary checking procedure was developed to alleviate this problem and is discussed in greater detail in the next chapter.

C. CONCLUDING REMARKS

Two very related methods have now been introduced for the segmentation of images. It can be seen tha the recursive splitting at hierarchical scope views is just a refinement of the basic MOCSE program with the program QUAD SPLIT using MOCSE as its major building block. A more detailed discussion is provided in the following chapter.

III. RECURSIVE SPLITTING WITH HIERARCHICAL VIEWS

In order to take infared data of a ship and process it into a useful output requires extensive preparation and knowledge of the system used. This chapter highlights the different problems involved in making these preparations and a more detailed look into the workings of the major procedures of the segmentation program used for this work.

A. GENERAL

The QUAD SPLIT program is the primary tool that was used in this research. The MOOSE algorithm is the basis for the QUAD SPLIT program. Therefore a thorough understanding of all the procedures of both was required. The version of the QUAD SPLIT utilized was written in PASCAL and contains approximately 5000 lines of code. Thus a more than basic knowledge of PASCAL was also required to ascertain what was happening in each of the many procedures and where all the information was stored for future use. With no prior knowledge of this language before the start of the research, very much valuable time was consumed in acquiring the needed knowledge as new complexities, structures, and formulations were encountered.

QUAD SPLIT is a very complicated program which was implemented without regard to speed of execution. This made the tracing through and understanding of the inner workings of the program very difficult because many of the procedures are nested within other procedures which again are nested in still other procedures.

The infared data available for evaluation and testing is on magnetic tape in a 64 X 256 array of gray scale levels

from 0 to 255. QUAD SPLIT requires an input data file of a 256 X 256 array. This data file was generated by formulating a short PASCAL program to copy the infared data in the first 64 lines of the array and putting the value 1 in the rest of the array. With this type of input file supplied to QUAD SPLIT, it required an average of 10 minutes for the program to run to completion and produce an output file.

B. MAJOR PROCEDURES

1. Hierarchical Segmentation

QUAD SPLIT takes the input data and reads it into a global record which is available throughout the program. The 256 X 256 array is considered the initial scope view and is at the highest level of the hierarchy at level 8 (2 ** 8 = 256). When a scope view is divided, it drops one level (i.e. 128 X 128 => level 7, 64 X 64 => level 6, etc.). The level 8 image is automatically divided into four quadrant scope views at level 7. The upper left-hand quadrant, called NW, scope view (from 0,0 to 128,128) is tested first. The histogram for this scope view is calculated which is then checked for significant peaks. The candidate peaks are then tested for spread width to determine if further division is required. If so, the level 7 scope view is divided into four level 6 quadrant scope views. The NW scope view is again checked first to determine whether further division is required. A minimum area criteria for scope view area precludes division below level 5 (32 X 32).

When a scope view requires no further division, the local minima between peaks is determined to set up threshold levels for assigning unique region numbers to the position in the scope view that fall within the threshold levels. For example, assume there are peaks in the histogram at gray

scale levels 100, 175, and 220with local minimas at 150 and 200. Then for each array position in this scope view with a gray scale level below 150, a unique region number would be assigned(i.e. 4), while array positions for gray scale levels between 150 and 200 would be assigned another unique region number (i.e. 132), and for array positions with gray scale levels greater than 200 would be assigned yet another region number (i.e. 777). Thus, three distinct regions within the scope view have been created. The scope view has now satisfied its initial tests and is considered terminal.

The next scope view to be evaluated is the upper right-hand, called NE, scope view on the same level as the last completed NW scope view. This continues on to the lower left-hand (SW) scope view and finally the lower right-hand (SE) scope view. For example, start at level 8 which automatically goes to level 7. The NW scope view of level 7 is checked first and assume no further division is required. The NE scope view of level 7 is then evluated, and assume this requires division to level 6. Here the NW scope view is evaluated first and assume it is terminal, followed by the NE scope view (also terminal), followed by the SW scope view (terminal), and finally the SE scope view which is assumed to require further division to level 5. Assume all four scope views here at level 5 are terminal, which causes the SE scope view at level 6 to become terminal, which completes the NE scope view at level 7 and causes it to be terminal. This same sequence is applied to the other scope views at level 7 and at any lower levels that may be required until all the scope views at level 7 are terminal which in turn causes level 8 to be terminal. This example is depicted in Fig. 3.1.

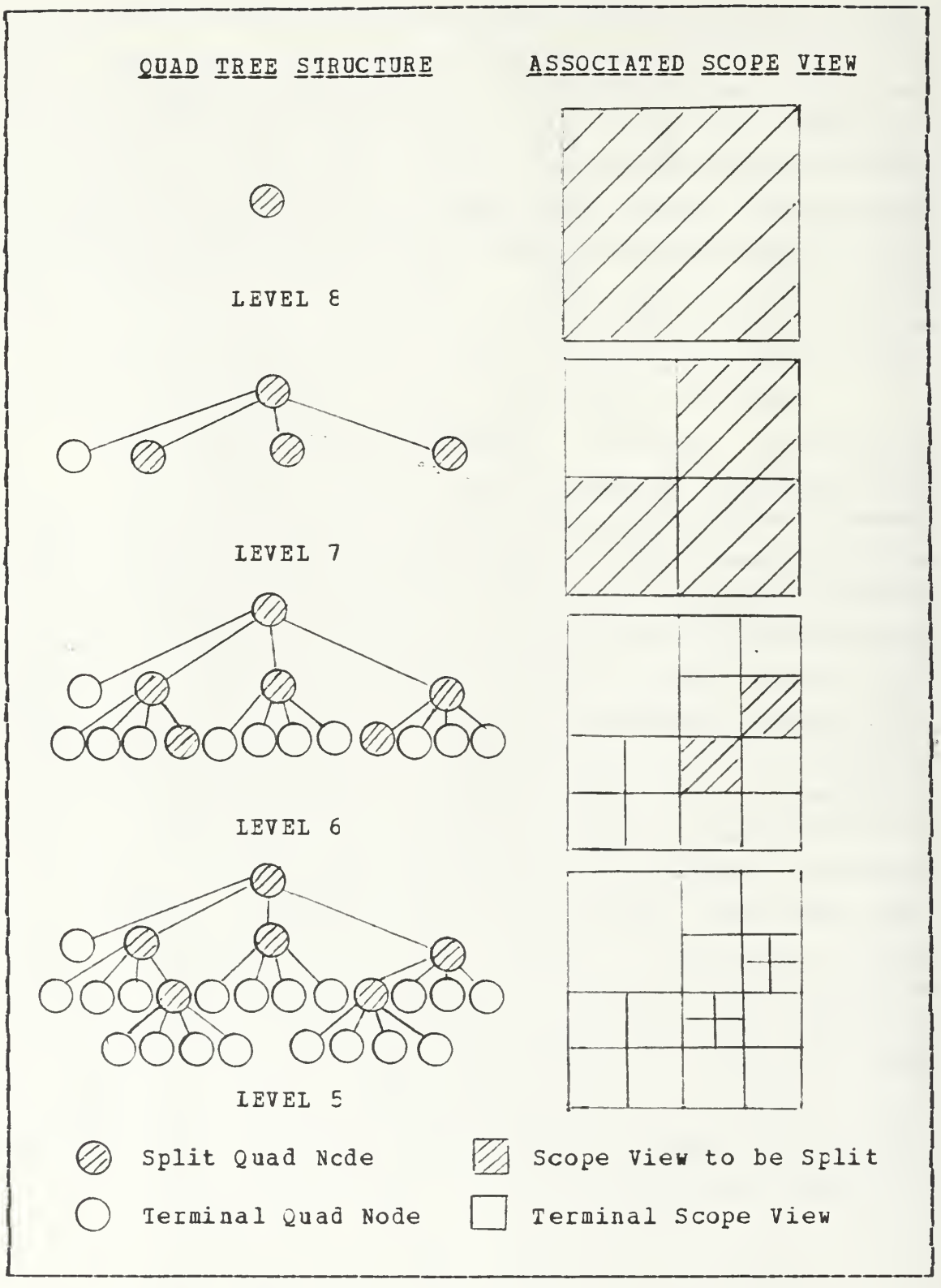


Figure 3.1 Quad Tree and Associated Scope View Developement.

2. Noise Elimination

The level 8 array now has a version which consists of the different region numbers. The number of times each region number occurs in the array gives the area for that specific region. This area is then compared to a constant which represents the minimum area a region must have in order not to be considered noise. If the region is less than this minimum area it is eliminated and is assigned the region number that surrounds it. The area for this surrounding region is then updated and the check continues until all regions have been checked.

3. Boundary Checking

The array of region numbers could now be made available in output to give a segmented representation of the original infared data. Problems may occur at the boundaries of the different scope views because the histogram and threshold levels used may be different in the adjacent scope views. Take for example the simple situations presented in Fig. 3.2. In scope view A, histograming and thresholding produce three regions that end at the boundary. In the adjacent scope view, scope view B, histogramming and thresholding may produce only two regions at the same boundary which may or may not coincide with the regions of the first scope view.

In scope view D there are two regions in which one region is partially enclosed by the other with an abrupt ending at the scope view boundary. On the adjacent boundary, scope view C, there are no breaks in the boundary due to different results for histogramming and threshold levels, even though it appears that there should be a continuation of the partially enclosed region of scope view D across the boundary.

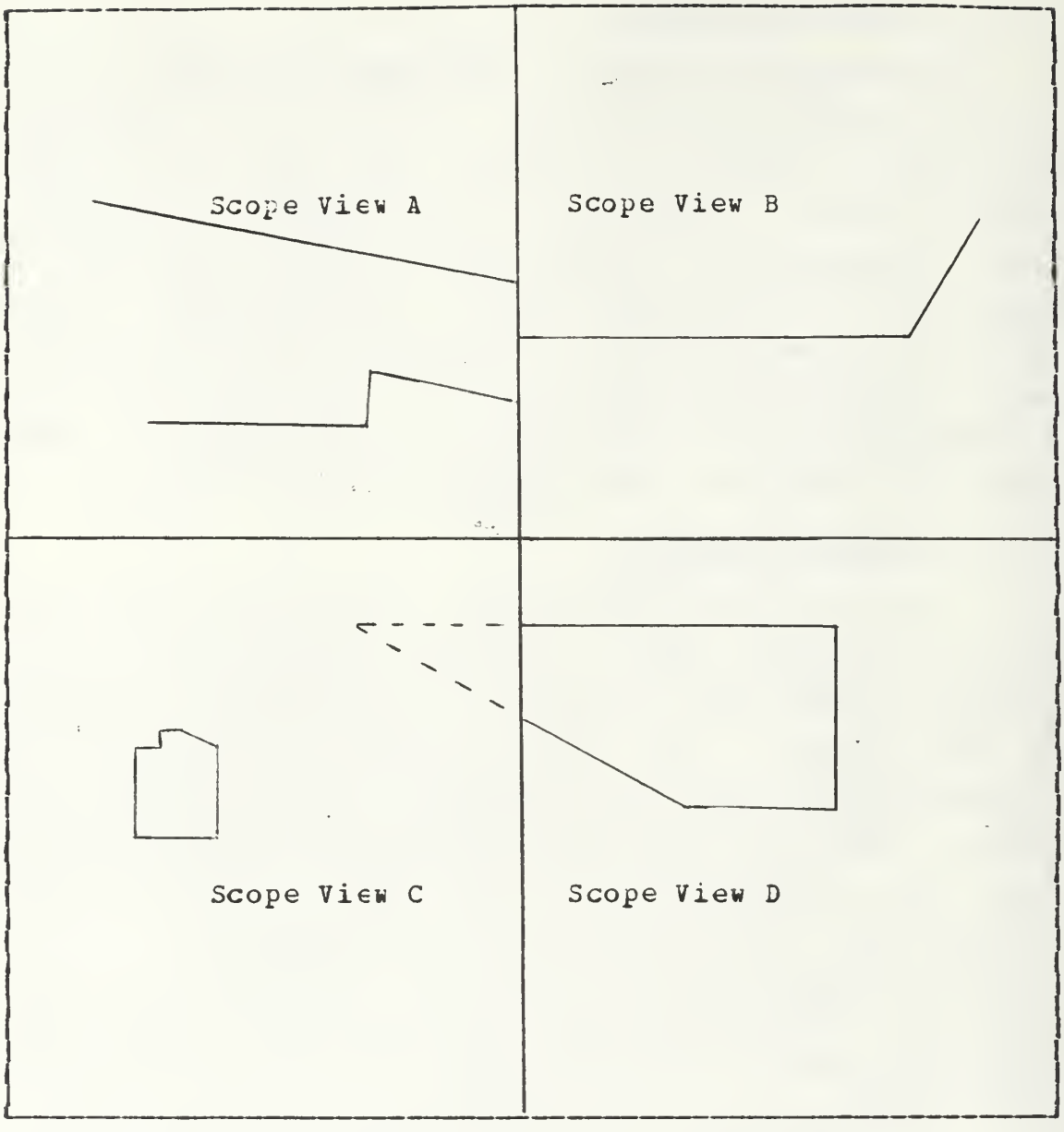


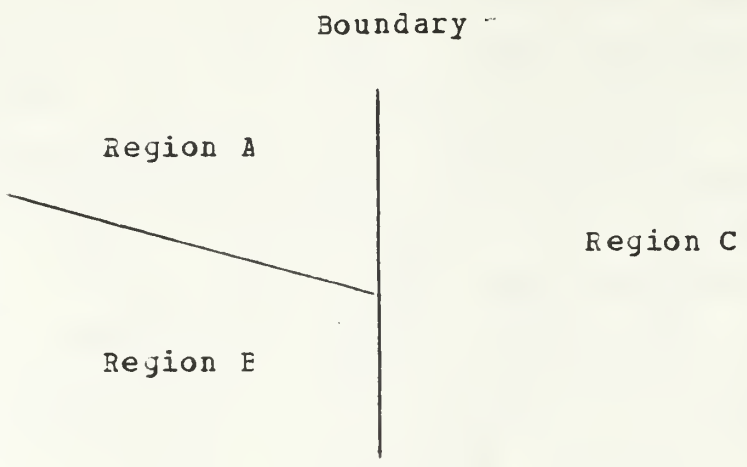
Figure 3.2 Boundary Checking Problems.

A procedure called SPLIT CHECK was thus incorporated into QUAD SPLIT to eliminate some of these potential problems. The scope views into which the image was hierarchically split is recalled. The four quadrant scope views at the lower levels are checked first. The rightmost column of

the NW and SW scope views, called the NS boundary because it is vertical and runs from top to bottom or north to south, is used to start the check. Each position of this column has its region number compared to the region number of the position just below it. If the two values are different then the boundary between two regions within the scope view has been detected and is called a break. When a break is detected, the row of the position where this occurred is remembered. The adjacent scope view boundary (leftmost column of the NE and SE scope views) is then checked within a range of +5 to -5 positions of the row in which the break was found. The check performed is the same as detailed above for finding a break. If a break is found within this range, then there is no problem for the desired result to be realized. If no break is found on this adjacent boundary, then the adjacent scope view is divided again, if possible, with histogramming, thresholding, and region number assignment again taking place. Boundary checking is then recommenced with the hope of now finding a break in the appropriate area on this adjacent boundary.

If this still fails, then the thresholds of the original scope view regions for which the break was found are recalled and averaged. If this new value falls within the peak interval for thresholding of the adjacent scope view region, then it is possible to dissect the peak interval into two new intervals and thus create two regions and a possible break. This is called threshold propagation and is depicted in Fig. 3.3 . If this still fails to produce a break, then nothing more is attempted and the boundary checking procedure continues.

This checking for a break is done at all positions down the column except at rows divisible by 32 where scope view boundaries exist and thus a break would occur. After this column is checked, its adjacent column is likewise

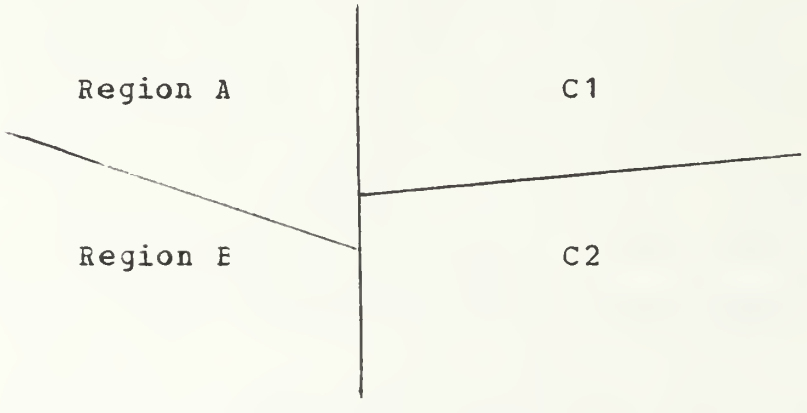


No Neighbor Break Exists

Assume that the peak interval of Region A >= the peak interval of Region B.

New threshold for region C = 1/2 (min. of interval A max. of interval B).

C1 > new threshold C2 < new threshold



Neighbor Break Now Exists

Figure 3.3 Threshold Propagation.

checked and compared with the original boundary column if a break is detected. Next the bottom most row of the NW and NE scope views, called the EW boundary is used to check for breaks in the column positions. Any breaks found are checked in likewise fashion across the boundary. This is again done for the other side of the boundary until all boundary positions inside the four scope views have been checked. This same procedure is repeated for all levels until the four scope views at level 7 have been finally checked, and the procedure ends.

4. Output Generation

Region numbers have now been assigned to all the regions in the different scope views and are available in a 256 X 256 array. To get a hard copy of this region map from a printer, the region numbers must first be converted to single letters, numbers, or symbols so that each array position representing the different regions can be represented by a single character. This is accomplished by using 55 different characters from the keyboard. The region numbers are first put through a modulo 55 function and the remainder is correlated to one of the 55 characters. This array of characters is then stored and is available for output to a printer in four 128 X 128 pieces.

By linking the QUAD SPLIT program with the PLOT 10 program available on the VAX-750, an output depicting the edges of all the regions can be displayed on the TEXTRONICS console.

IV. IMAGE RESTRUCTURING

A. LIMITATIONS OF QUAD SPLIT

1. Region Numbering

As detailed in the previous chapter, each region in the scope view has its own region number which correlates to its own region character in the hardcopy output. At boundaries there will be different characters, even though both sides of the boundary may have the same gray scale level in the original infared data. Therefore background features such as sky, horizon, water, etc., are broken up and displayed by many characters depending on how many times the image has been split over that area. This problem is somewhat eliminated by useing the PLOT 10 program in some other form of output because only the edges are displayed. The output must then be visually interpreted to find the desired features.

2. Region Map Output

Problems of duplicate characters can arise in the output because of using the modulo 55 function to operate on the region numbers when assigning characters to the listing output. In QUAD SPLIT the maximum number of regions that can be generated is limited to 64, while region numbers can have values from 1 to 4096. Thus duplication of characters is highly likely, but their occurence in adjacent regions or across boundaries is not often.

3. Background Features

The procedures and tests incorporated into QUAD SPLIT for the calculation of the area and the centroid of

each individual region, and for making a selection of good features are limited because of the region propagation problem. Background features that may occur in the image could have been broken up at least at scope view boundaries. Thus, area and centroid information become inaccurate for selection of a target from the background regions. Additional procedures for more extensive geometry comparison is required to distinguish a target from the rest of the regions.

B. ALGORITHM FORMULATION

1. Region Number Propagation

Various ideas were considered to accomplish the desired region number propagation across the boundaries. One method considered was to count the number of times a region number occured at a boundary and compare the result with a similar count on the opposite side of the boundary. The assignment of the same region number to those that occured almost an equal number of times on both sides of the boundary would then take place. This was quickly rejected because the count on one side of the boundary may occur at the top while on the opposite side of the boundary it could occur at the bottom, and thus the same region number could be assigned to completely unrelated areas.

Another idea was to count the number of times two pairs of region numbers occured across a boundary and to assign the same region number to the pairs that occured most frequently or that exceeded a set minimum count. This was also rejected because one side of a boundary may have only one or two region numbers while the opposite side of the boundary could have numerous region numbers and when counted in pairs could all realize the same result. Then which region numbers should propagate and which should not would have to be decided.

The idea that was finally selected as the most likely to produce the desired result was to determine where breaks were encountered on both sides of the boundary and then assign the same region numbers to the pair of regions that occur across the boundary on either side of the break. If a break is detected on one side of the boundary but with no adjacent boundary break found within a specified range, then all regions would retain their present region numbers. This is the basis upon which the region number propagation algorithm will be implemented.

2. New Region Map

Since there are a maximum of 64 regions available from the QUAD SPLIT program, this implies that there are at most only 64 different region numbers utilized. Thus, by adding nine more characters to the list of 55 characters now available, all the 64 different regions could be assigned a unique character corresponding to its region number without duplication being a problem.

3. Feature Tests

Basic FORTRAN procedures for calculating area, perimeter, center of gravity (or centroid), size, and compactness were known to be available on the VAX-750 system by using the SPIDER programs [Ref. 4]. These procedures would use the information generated from the region number propagation procedure to produce their respective results for each region. The retrieval of the desired targets could then be based on this information.

C. ALGORITHM IMPLEMENTATION

1. Background Region Check

The implementation of the algorithm for region number propagation was based on the boundary check procedure of the QUAD SPLIT program. Since it was to help detect the background portions of the image, the procedure was named Background Region Check , or BAKREG_CK for short. The procedure was to read the region number information from QUAD SPLIT into a new array which then can be changed without affecting the original record.

The same type of boundary checking as discussed previously was utilized to detect a break. If a break was detected on the opposite side of the boundary within +5 to -5 positions of the original break, then the pair of two adjacent regions above (or to the left) of the break would be assigned the region number of the region to the left for NS boundaries (or the region on top for EW boundaries), and the pair of adjacent regions on the other side of the break would be assigned the region number of the region below and to the left of the break for NS boundaries (or above and to the right of the break for EW boundaries). A pictorial diagram of the above situation is given as Fig. 4.1. If no break is found in the adjacent boundary, then no changes are made.

One potential problem that was realized at this point was the situation where multiple breaks on one side of the boundary were matched by offset multiple breaks on the adjacent boundary within the +5 to -5 position range. For example, (see Fig. 4.2), assume we are checking column 64 and a break is encountered at row 15, with the region above the break designated region 'A' and the region below the break region 'C'. A corresponding break on the opposite side of the boundary (column 65) occurs at row 19, which is

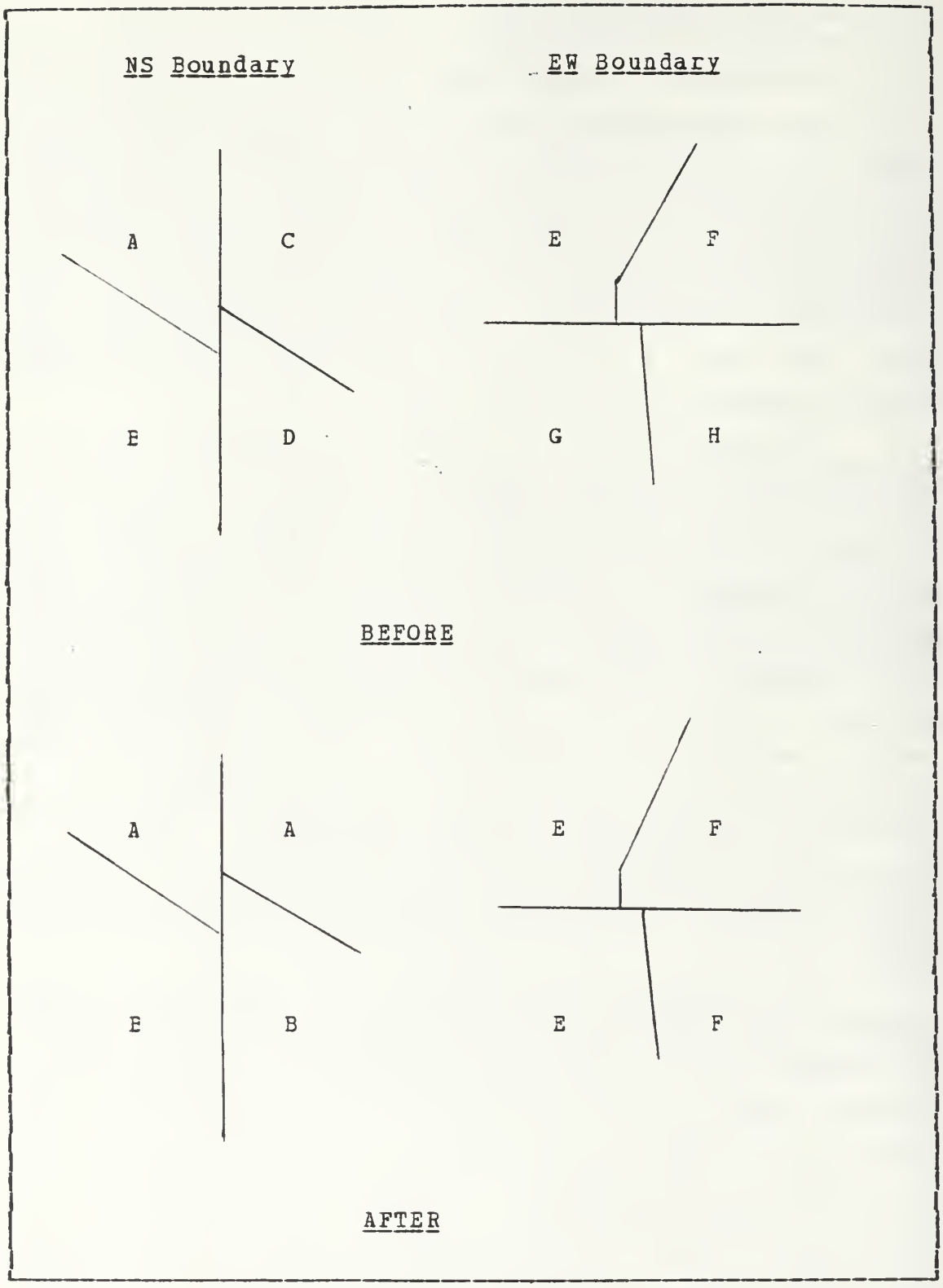


Figure 4.1 Example of Boundary Checking.

within the +5 to -5 position range. Call the region above this break region 'B' and the region below as region 'D'. Since corresponding breaks on both sides of the boundary have been found, then region propagation occurs with regions 'A' and 'B' being assigned region number 'A', and regions 'C' and 'D' being assigned region number 'C'. Checking now continues down column 64 for another break and assume for this example that it occurs at row 23 with region 'C' above the break and region 'E' below. Checking adjacent column 65 would show a break at row 19 again since it is still within +5 to -5 positions of this new break in column 64, while an additional break in column 65 at row 24 would be missed.

To alleviate this problem additional steps were incorporated so that once two adjacent breaks have been used for region number propagation, then neither can be used again by additional breaks. Also it was noted that only one pass along a given set of adjacent boundaries is sufficient for this check because once one boundary is checked and compared to its adjacent boundary, then all of the possible connected regions will have been recognized and no further information would be gained by checking the adjacent boundary against the original boundary.

2. Region Map Output

The assignment of new region numbers from 1 to 64 was implemented by a reiterative routine which would run from 1 to 4096 checking the region number array at each position. When an original region number was found, it was reassigned a new region number starting with 1 and going up to the maximum number of regions utilized. Each time a position was reassigned a new region number a counter would be incremented and this count was compared to the maximum number of postions available (65536) and would stop the iteration once this was reached.

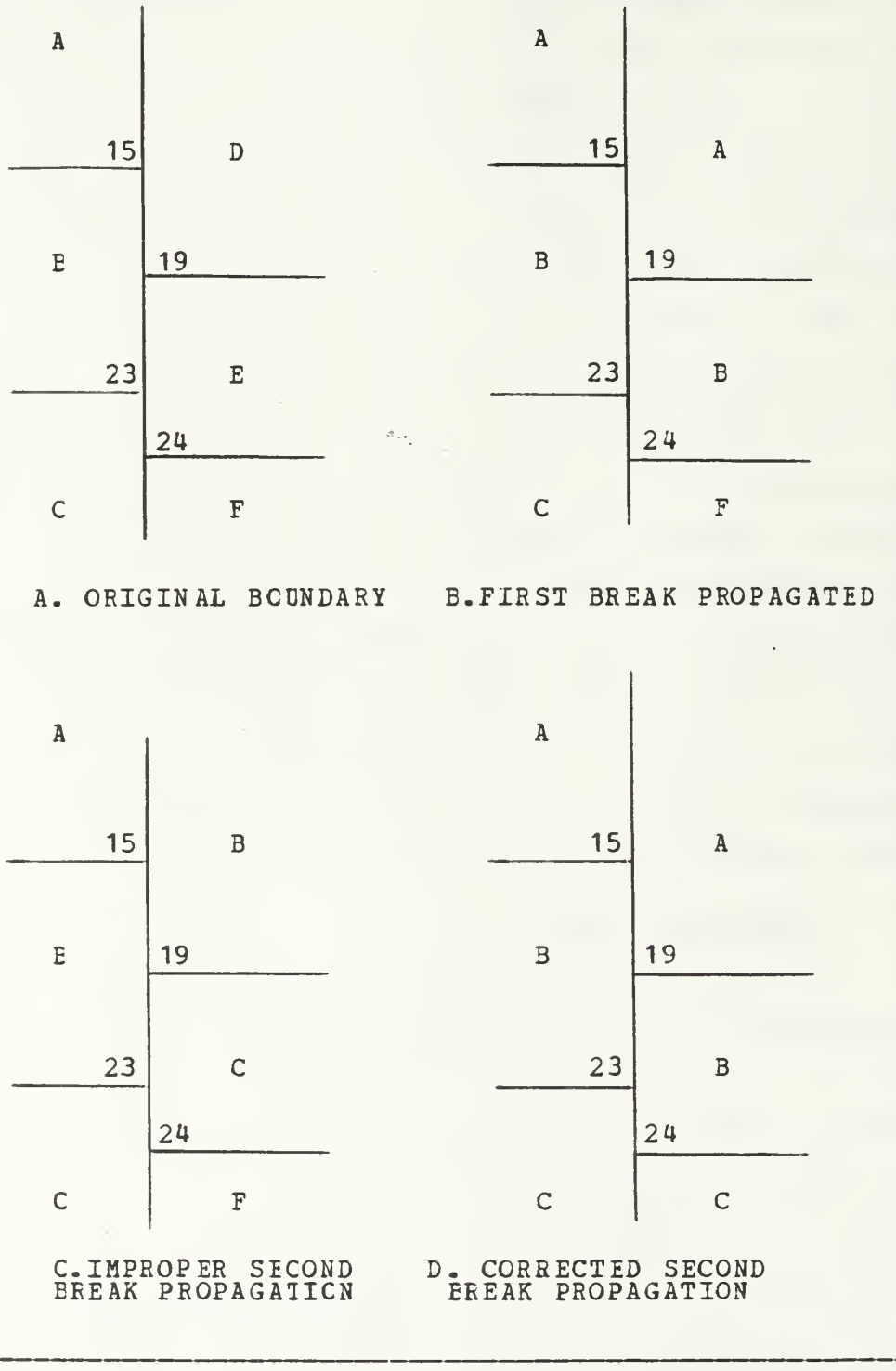


Figure 4.2 Boundary Checking Example for Multiple Breaks.

To simplify processing throughout this set of procedures, this reassignment of region numbers was incorporated with the inputing of the information from the QUAD SPLIT region number record.

An additional nine characters was also added to the list of characters utilized to represent the region numbers. The characters were then assigned to respective region numbers after the region number propagation procedure was completed to give a hardcopy output available in four 128 X 128 segments.

3. Geometry

The procedures for area, perimeter, and center of gravity (centroid) were the only SPIDER procedures utilized. The procedures for size and compactness both called the area and perimeter procedures and then applied a simple equation to produce their respective outputs. Since the area and perimeter outputs were already available, the following equations were used to directly calculate the size and compactness:

size = (2 X area) / perimeter

compactness = (4 X PI X area) / perimeter

D. CONCLUDING REMARKS

The program has now been thoroughly reviewed and new procedures have been formulated and implemented. The procedures as written are provided as Appendices A thru E. The whole package must now be brought together so that test runs with real data can be observed. The testing and evaluation of the program is the subject of the next chapter.

V. TEST AND EVALUATION

The basic ideas behind all of the programs have been looked over extensively and the algorithm changes to these programs are implemented. The testing of all procedures with the infared image data is the subject of this chapter. The evaluation of the test data and conclusions reached are in the final sections.

A. EXPERIMENTAL RESULTS

Testing was an on going effort during the entire research period. As each small program or procedure was written, it would require testing to verify that it gave the desired results. This section will only cover the testing of the major programs and procedures that have been discussed in detail in prior chapters.

1. QUAD SPLIT

The QUAD SPLIT program had been extensively tested prior to this work and the limitations that were encountered are documented in Chapter IV. The average running time of the program for the infared image data was approximately 10 minutes. Fig. 5.1 is an example of a portion of the segmented output listing. This example makes it obvious that the output is very disjointed at the scope view boundary occuring at column 128. It would be very difficult to gather any useful information from the output in this form.

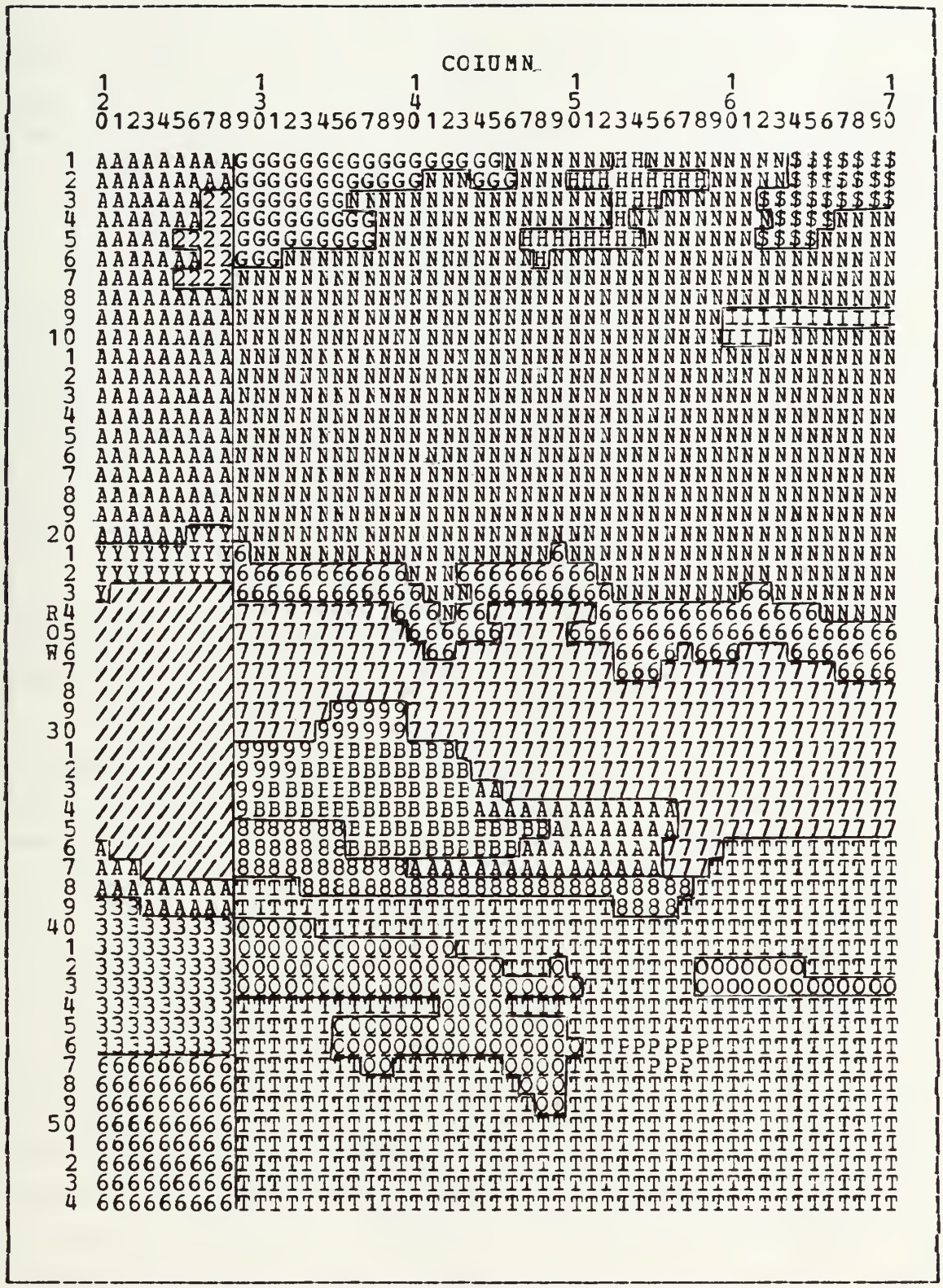


Figure 5.1 An Example of QUAD SPLIT Region Map Output.

2. Background Region Check

The new procedures added to the QUAD SPLIT program are the real essence for the testing conducted in this work. As a new procedure was added to the original program, it was tested to check if any further refinements would be required.

a. Region Number Propagation

The first portion of the background region checking procedure to be implemented was the region propagation algorithm and the associated new region map output listing. The initial tests of these procedures led to the detection of the problem where the same region symbol was being utilized more than once in the same region map. After the incorporation of additional steps to rectify this problem, a series of tests was performed using five different infared images as the input data. The total time for each program to run was approximately 16 minutes, with the fastest completed in 10 minutes and the slowest in 21 minutes. The time required for each run was determined by how many times the image data was split and by how many regions were formed. The data with larger targets had less regions and therefore ran the fastest. The same portion of segmented output listing as presented in Fig.5.1 but with region number propagation in effect, is given in Fig.5.2. As can be seen, the scope view boundary at column 128 has been virtually eliminated and background features, such as regions 'B, M, and J', which propagate across the entire output section are easy to pick out. For each of the five different infared images used, successful region number propagation was observed. The background region map outputs are presented as Fig.5.3, Fig.5.4, Fig.5.5, Fig.5.6, Fig.5.7, Fig.5.8, Fig.5.9, Fig.5.10, Fig.5.11, and Fig.5.12.

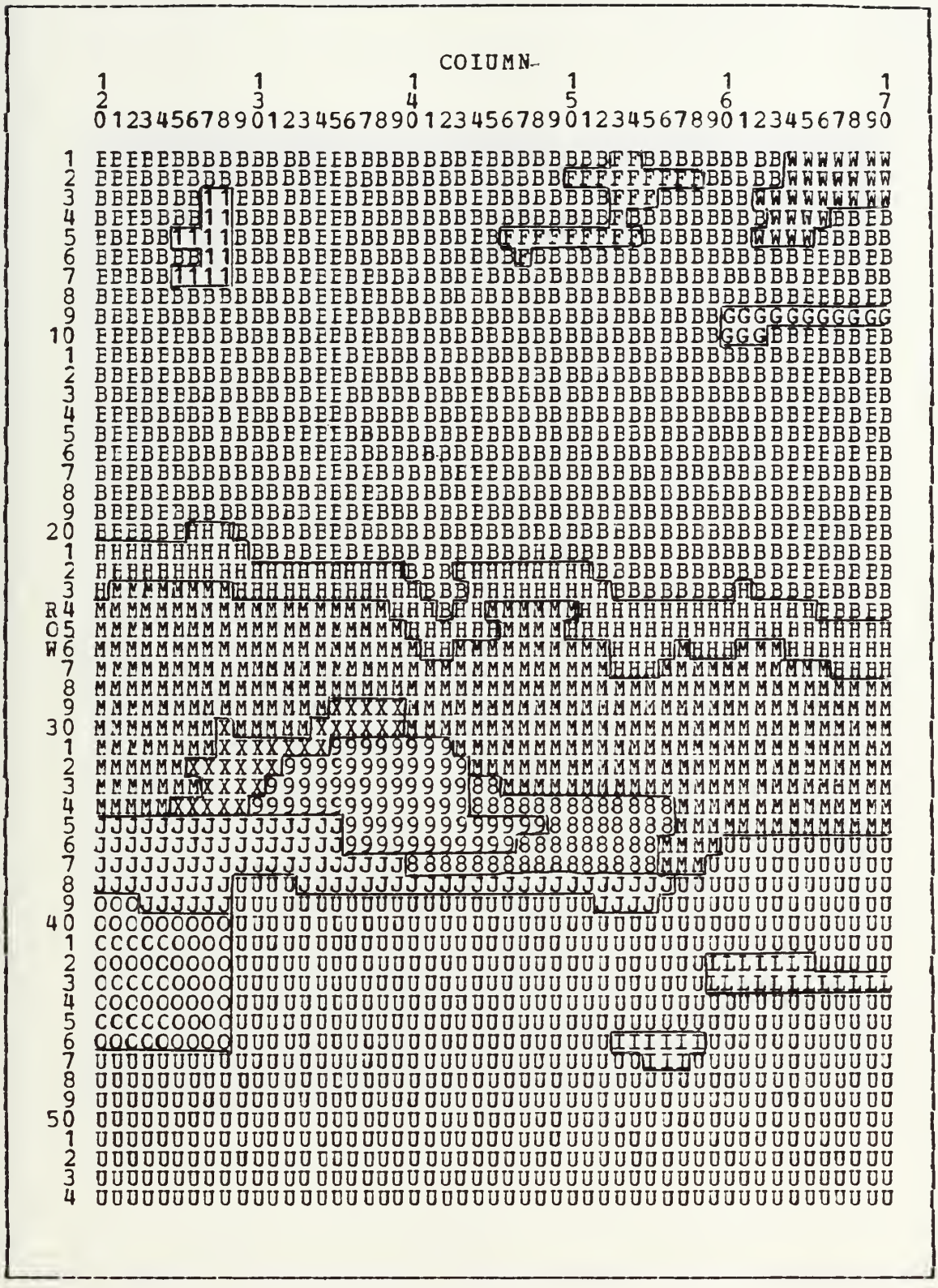


Figure 5.2 Region Map Output with Region Number Propagation.

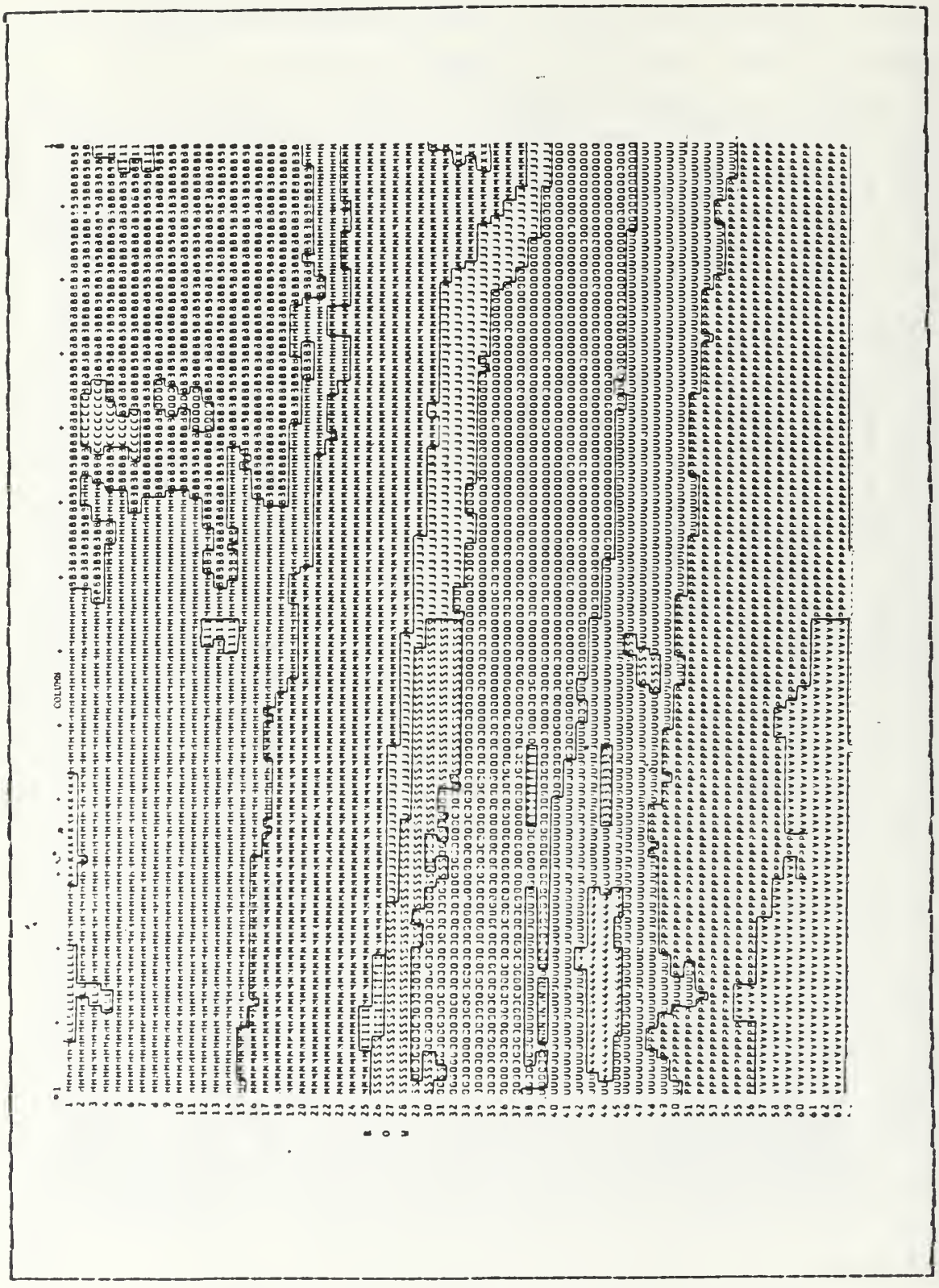


Figure 5.3 Image A Background Region Map--col. 1 to 128.

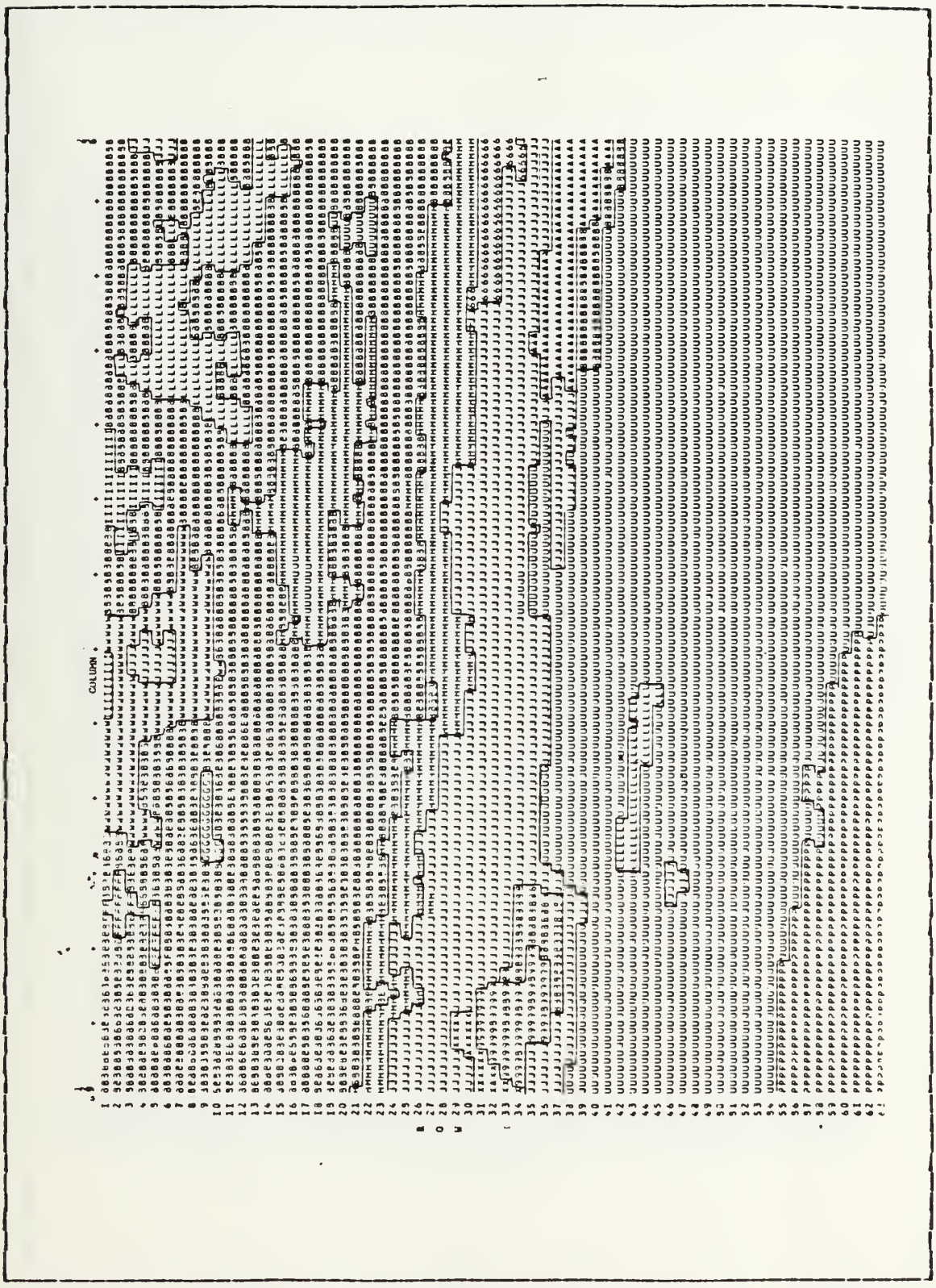


Figure 5.4 Image A Background Region Map--col. 129 to 256.

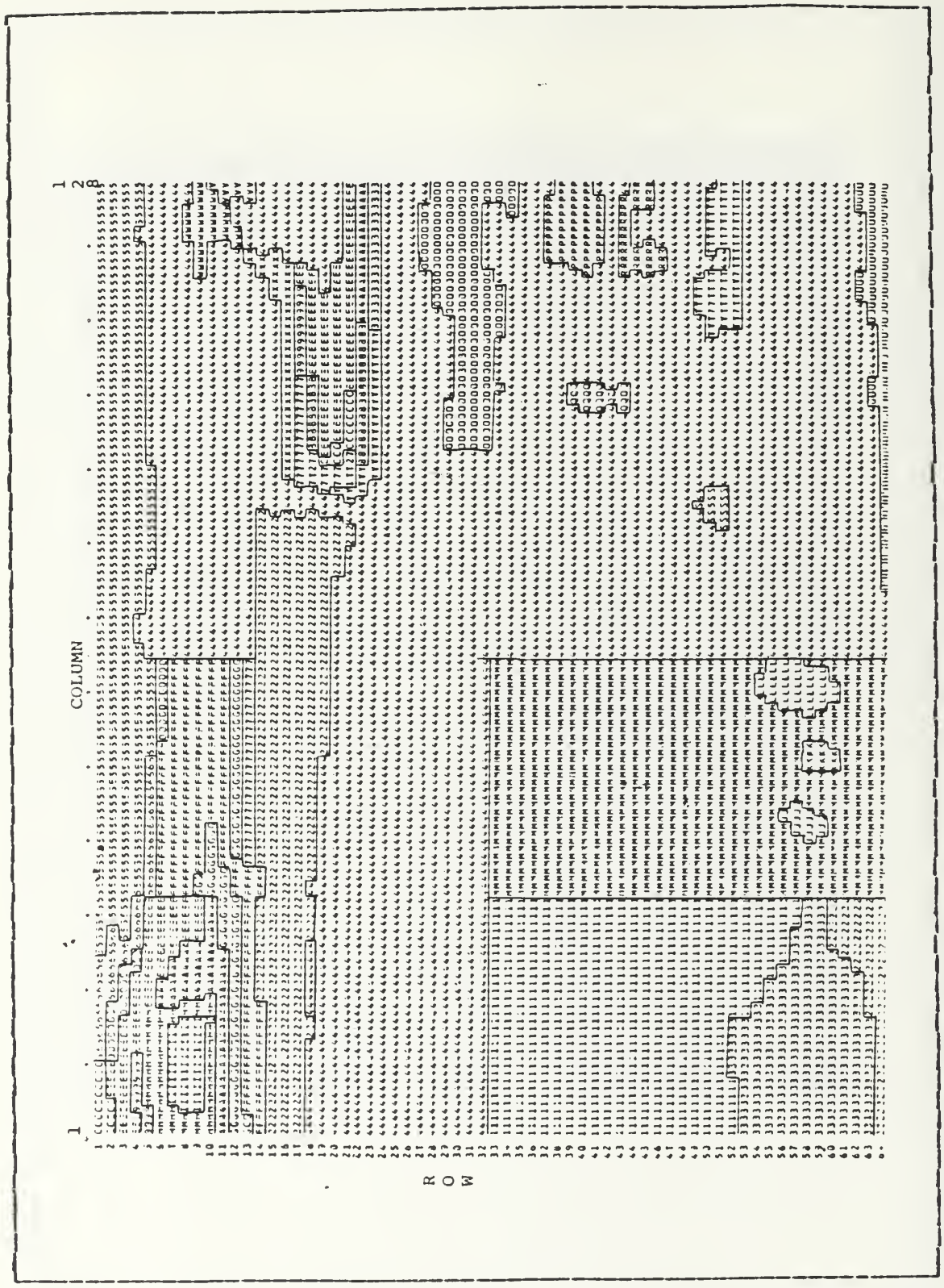


Figure 5.5 Image B Background Region Map--col. 1 to 128.

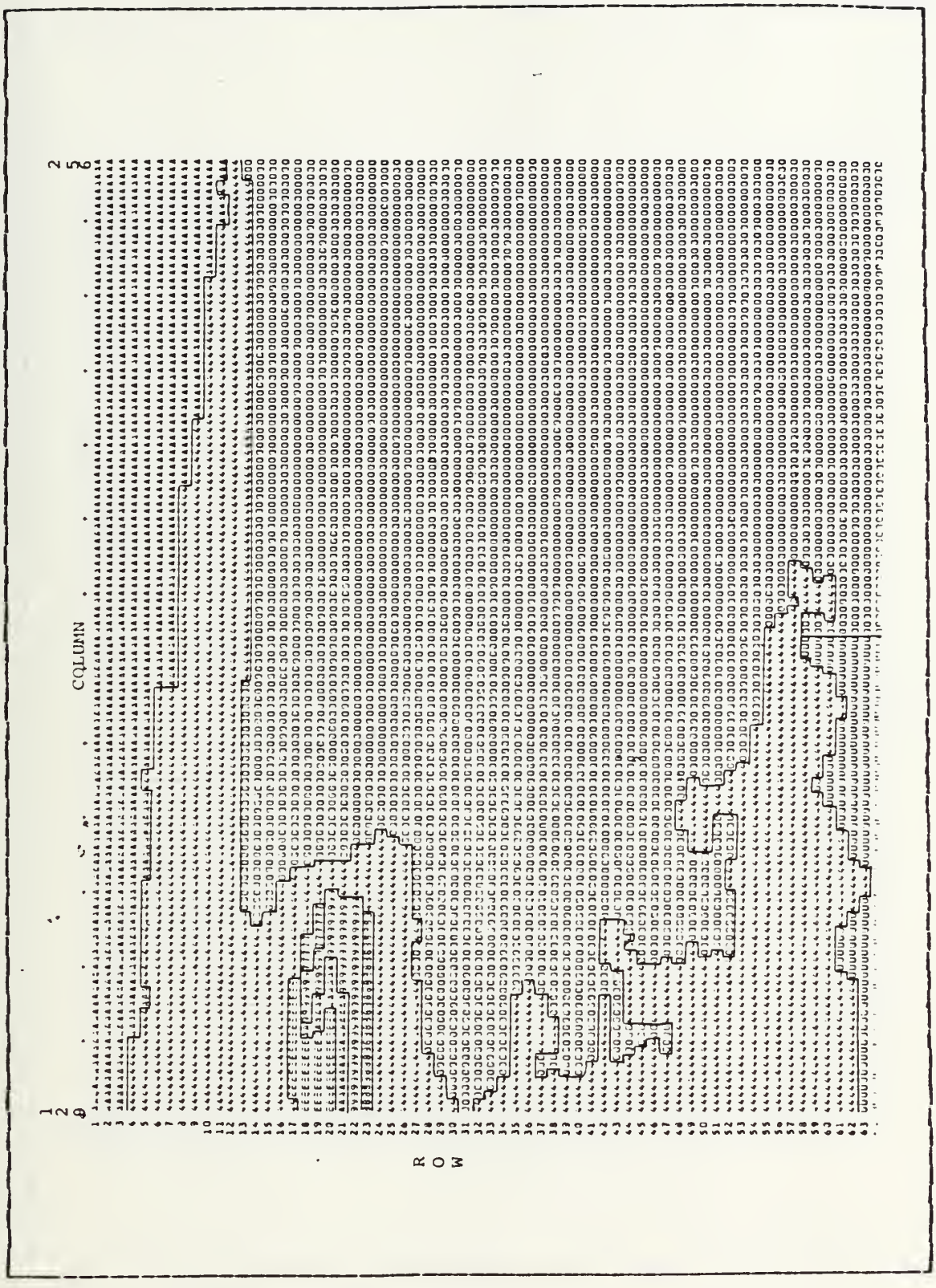


Figure 5.6 Image B Background Region Map--col. 129 to 256.

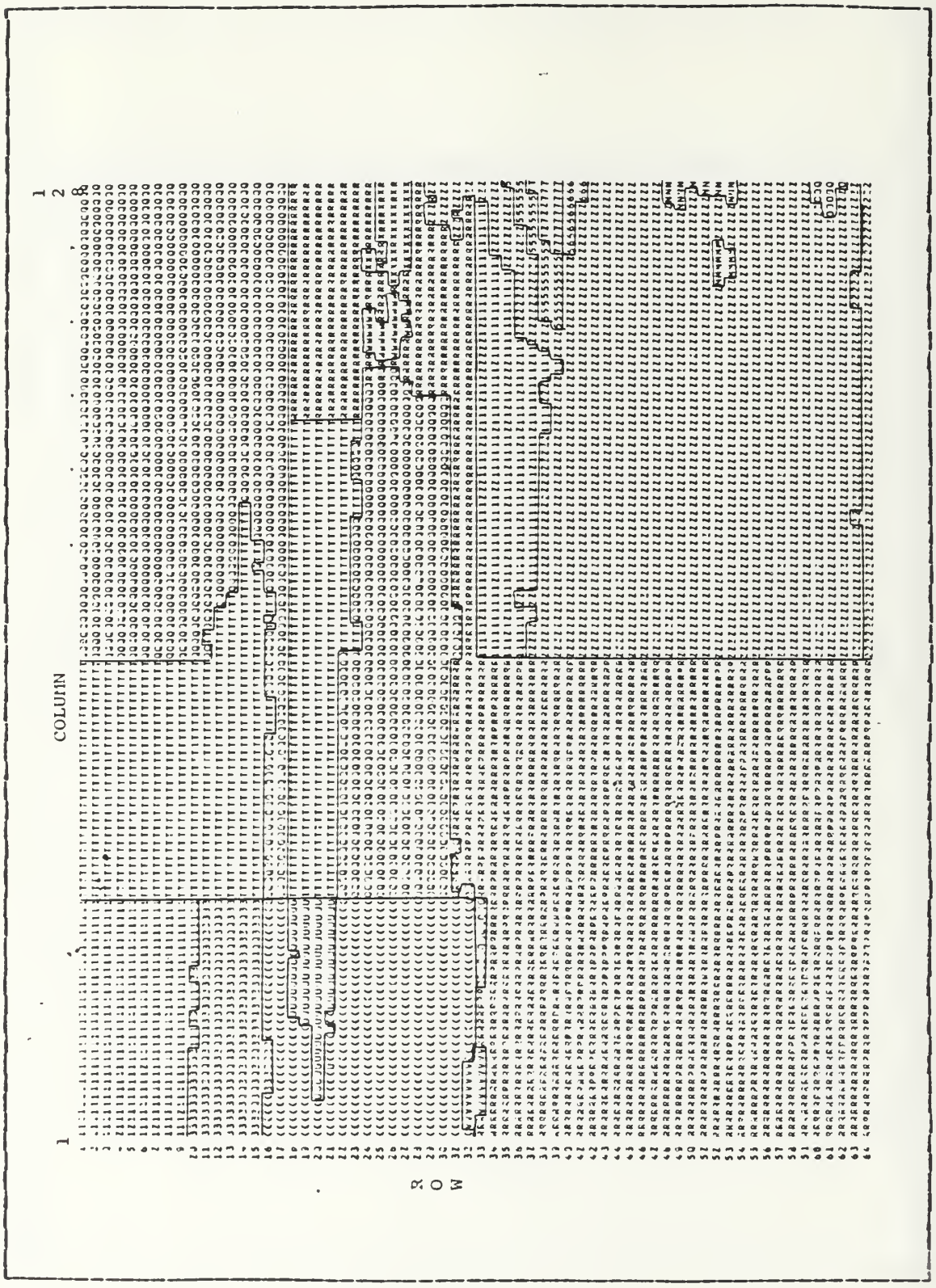


Figure 5.7 Image C Background Region Map--col. 1 to 128.

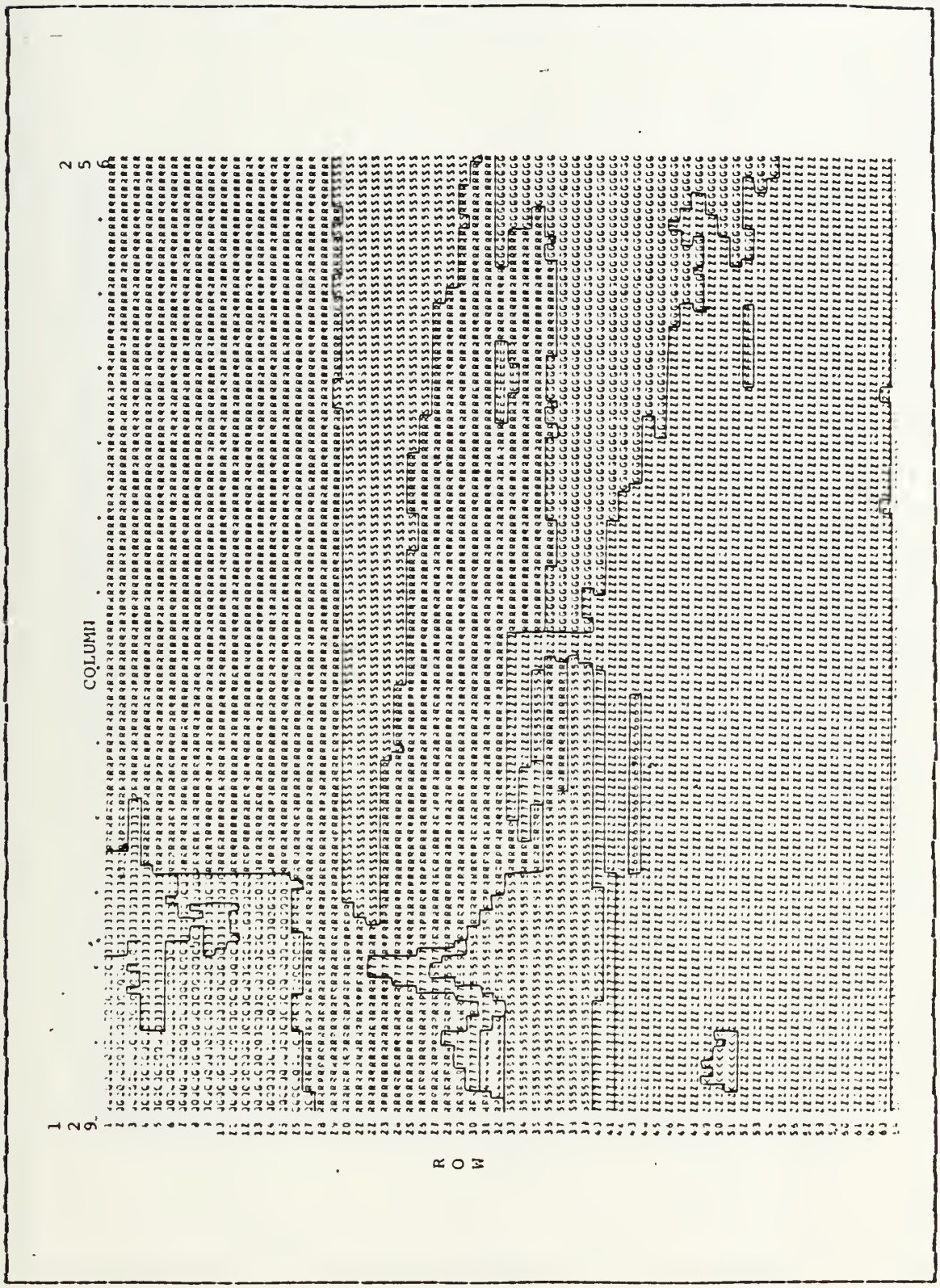


Figure 5.8 Image C Background Region Map--col. 129 to 256.

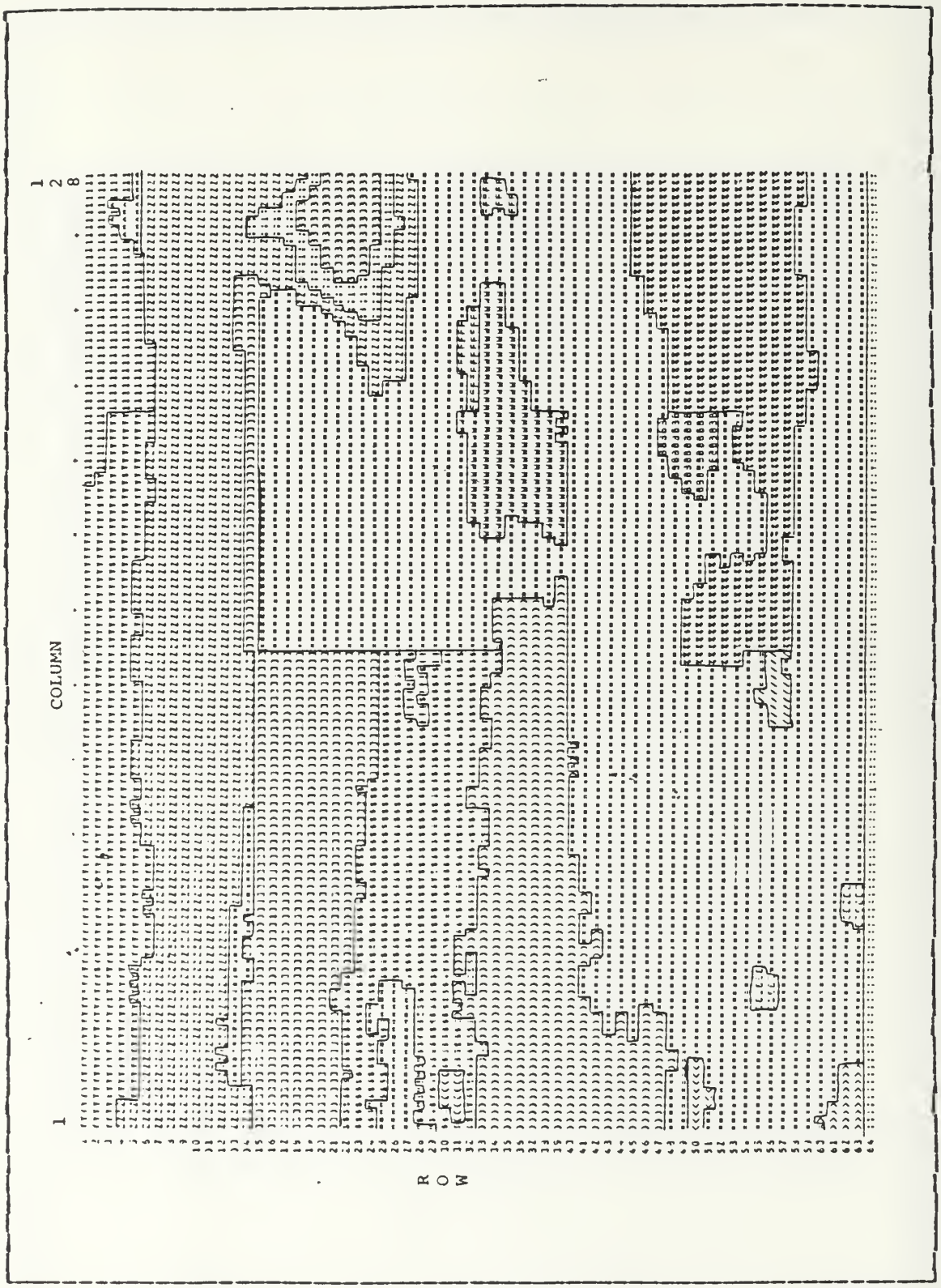


Figure 5.9 Image D Background Region Map--col. 1 to 128.

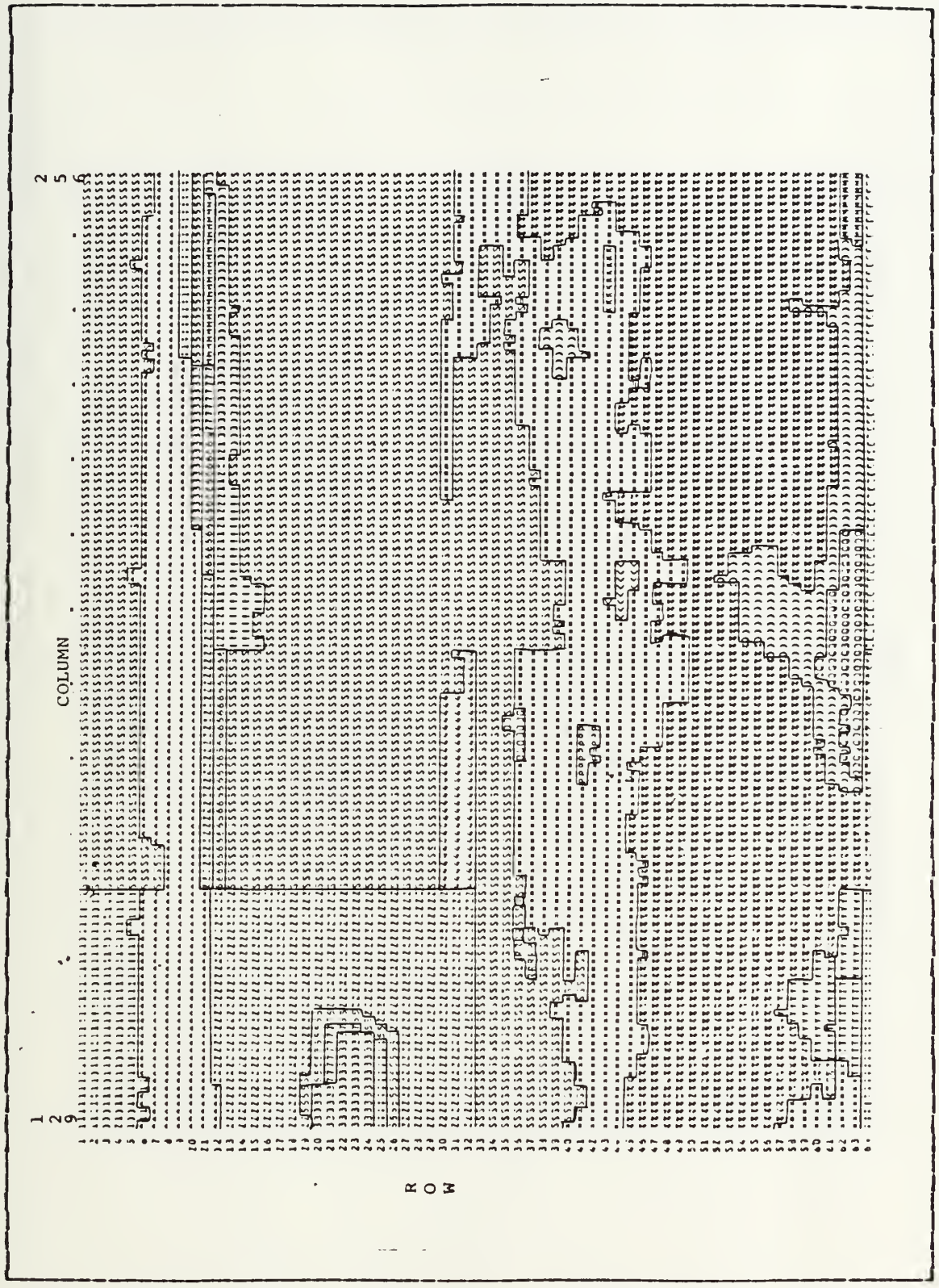


Figure 5.10 Image D Background Region Map--col. 129 to 256.

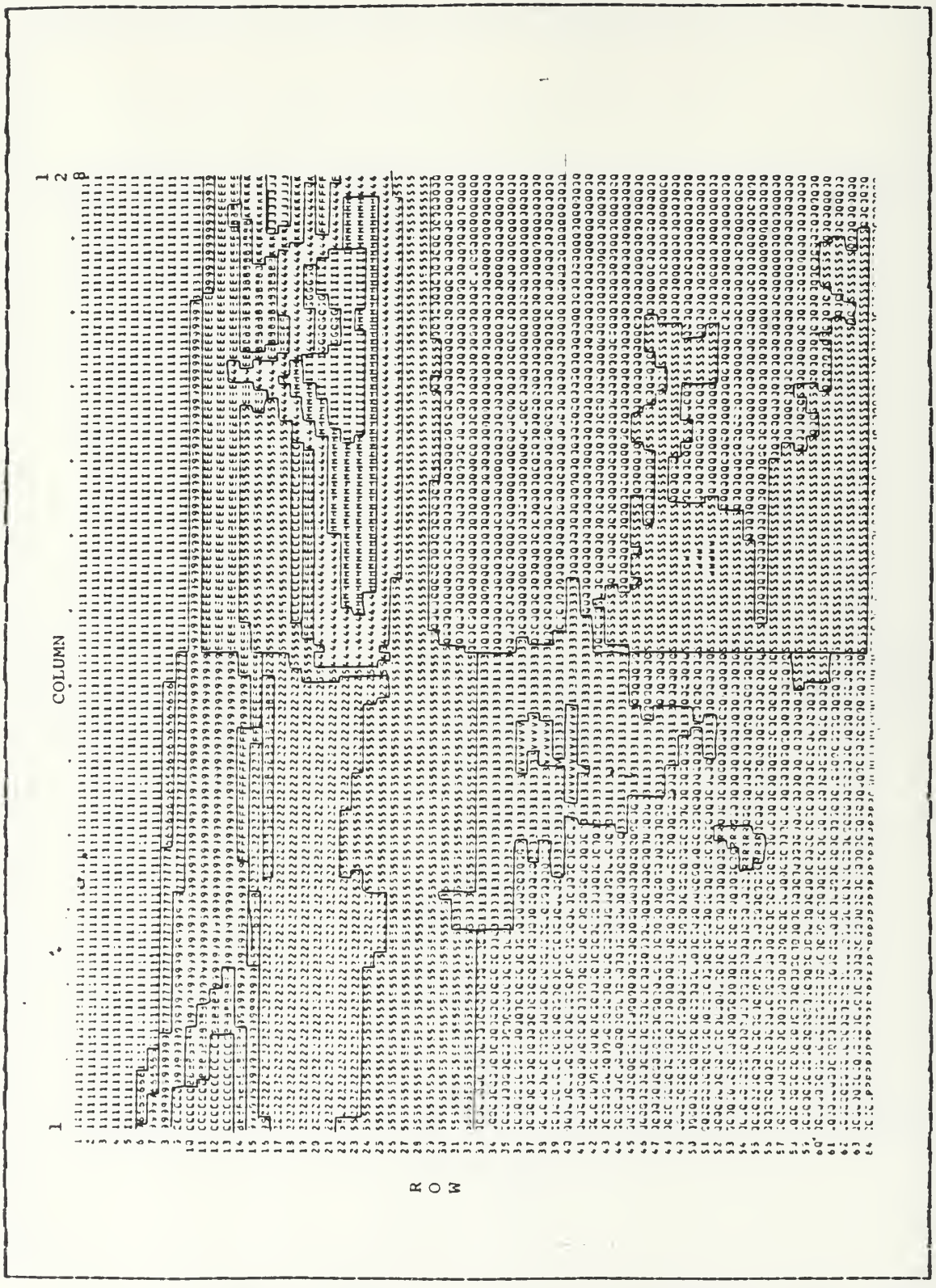


Figure 5.11 Image E Background Region Map--col. 1 to 128.

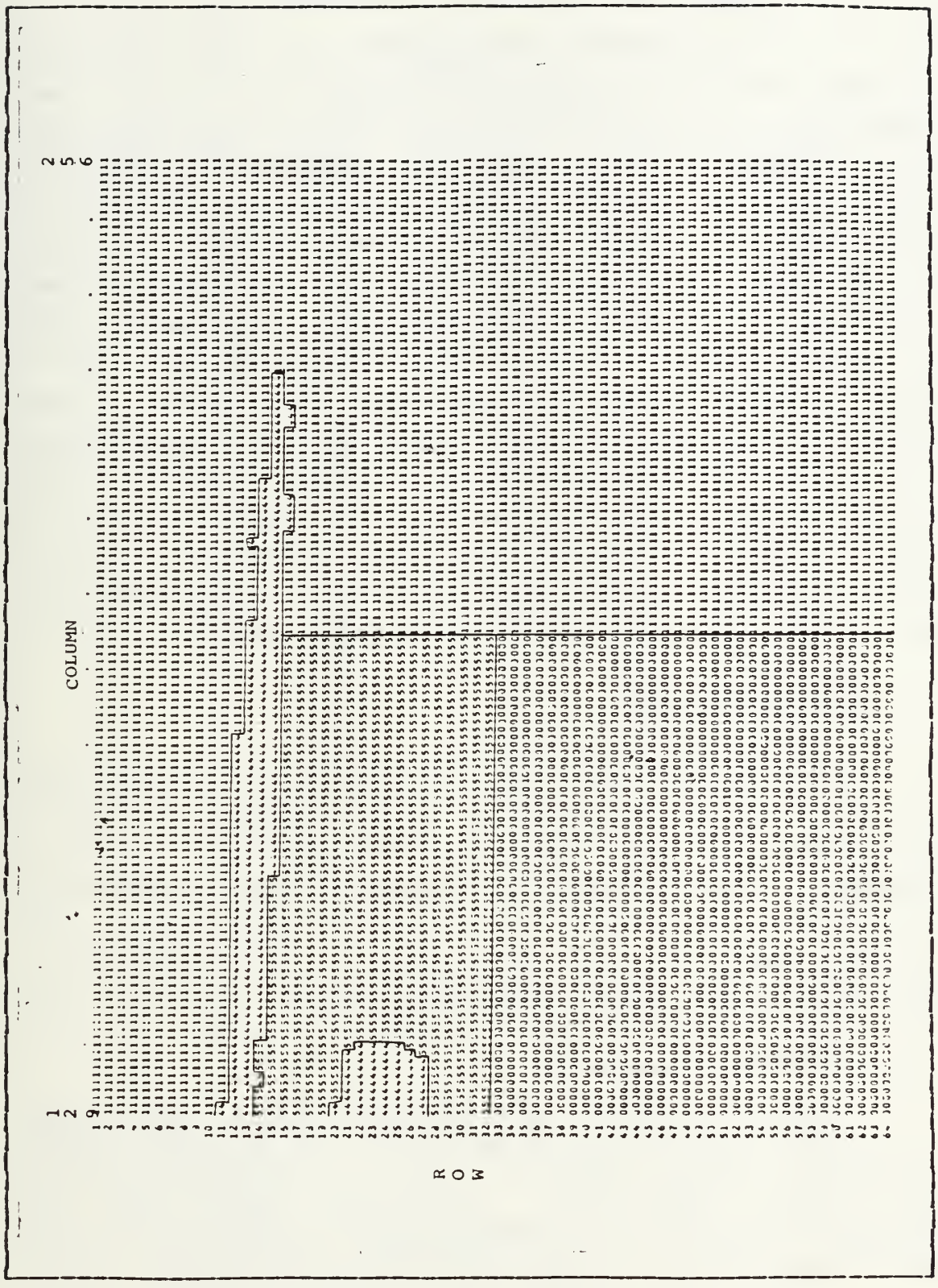


Figure 5.12 Image E Background Region Map--col. 129 to 256.

b. Geometry

The procedure for obtaining the area, perimeter, centroid, size, and compactness for each of the regions was tested next. Each of the individual FORTRAN procedures were tested separately by short routines with dummy input data to ensure that the desired result would be obtained and to verify that all required parameters were supplied correctly. Each of these were tested satisfactorily and then the area, perimeter, and centroid procedures were incorporated into the basic program. Since size and compactness can be derived from the area and perimeter procedures, it was decided to save some calculating time by using the equations to generate the desired result. A sample listing of the data generated for the regions shown in Fig.5.2 is given in Table 1. All five infared images were processed with these added procedures and less than one minute of processing time was added to the overall run time respectively.

B. EVALUATION OF DATA

All of the tests have now been completed. The data must now be evaluated to determine if the desired results of picking out the target from the background and noise are achieved. Time limitations precluded the generation of procedures to incorporate the program for this purpose. The following evaluation was conducted by visual examination of the data and some manual calculations.

1. Background Regions

After reviewing the background region map outputs, it could be readily observed that the background sections were clearly recognizable. They were regions that went generally across many boundaries and contained a major portion of the area. From the Geometry data files, it was

TABLE 1

Sample Geometry Data

REGION SYMBOL	AREA	PERI-METER	CENTROID ROW	CENTROID COL	SIZE	COMPACT-NESS
1	26	18	8	97	2.89	1.008
8	48	20	35	148	4.80	0.513
9	70	46	33	138	3.04	0.416
B	3275	848	12	167	7.72	0.057
F	25	42	3	152	1.19	0.178
G	15	24	9	164	1.25	0.327
H	1981	752	14	95	5.27	6.044
I	9	16	46	156	1.12	0.418
J	1198	22	31	160	109	31.100
L	34	32	43	172	2.13	0.417
M	1234	334	23	94	7.39	0.139
O	1230	356	37	65	6.91	0.122
U	3815	16	48	164	477	187.30
W	190	114	4	182	3.33	0.184
X	35	42	31	131	1.67	0.249

noted that the regions with the largest areas and size correlated with the background regions noted visually in all five cases. Therefore, all of these regions could be easily eliminated in a search for the good target. In the example of sample data given in Fig.5.2 and Table 1 the regions with symbols 'B, H, J, M, O, and U' would then be eliminated. For the five infared images evaluated, this type of background region determination would eliminate one third of the regions and approximately ninty percent of the area.

2. Size vs Compactness

A correlation between size and compactness was then attempted for the remaining regions. From all the data obtained, no useful correlation could be obtained. Even after looking at the background region map outputs and selecting likely candidates to check for any correlation, none could be found that would cover the target and get rid of the other regions. Using the output map of Fig.5.2 as an illustration, the regions with symbols '8, 9, and X' seem to be the most likely candidates for being the target representing the ship. Checking the size and compactness data for these three regions from Table 1 yields a size range of 1.67 to 4.80 and compactness range of 0.249 to 0.513. Two other regions fall within this size range, three more within the compactness range, and one region falls within both ranges. Similar results were observed for the other four images. More information is therefore required to make a good decision.

3. Dimensions of Regions

The dimensions of the regions were not incorporated in any of the procedures for the program under test. After reviewing all the background region maps, it was noticed that there were numerous long and thin regions. These regions were noted to be usually close to boundary areas and are generally caused by some noise or distortion in the infared image. If these regions were eliminated, an additional twenty five percent of the total regions could be removed.

The dimensions for the regions remaining after background region elimination for the example sample were obtained from the region map output and the results are tabulated in Table 2 . By reviewing all the data from the

five infared images, it was noted that if any dimension was of length 3 or less, or if the column lenth divided by row length did not fall within a range of 0.1666 to 6, then these regions could be assumed to be noise and eliminated. Therefore, from Table 2 data, region symbols 'G, I, and L' could be eliminated.

TABLE 2

Sample Region Dimensions

REGICN SYMBOL	ROW NUMBER MIN	MAX	LENGTH	COLUMN NUMBER MIN	MAX	LENGTH
1	3	7	5	125	128	4
8	33	37	5	140	156	17
9	31	36	5	130	148	19
F	1	5	5	146	158	13
G	9	10	2	159	170	12
I	46	47	2	154	159	6
I	42	45	4	159	183	25
W	1	9	9	162	204	42
X	30	34	5	125	139	15

Another use cf the dimensions of the regions could be to use them in ccnjunction with the centroid data to locate regions which are in close proximity to each other. The infared data of a ship usually breaks the ship up into different regions because of the difference in intensity cf the infared radiaticn given off by different areas of the ship. These regions would all be close together and this

fact could aid in the selection of the good targets. A simple test was devised for taking the regions that are remaining and checking to find out if both the row and column values of their respective centroids are within a given fixed value. For the five infared images tested, a value of ten seemed tc work satisfactorily. For the sample data cf Table 1 this test would be satisfied for region symbols '8 and 9' and '9 and X'. Therefore, a good choice for the good regions representing the ship would be the regions designated by symbols '8, 9, and X', which were the same regions selected for being possibly good targets from the background region map of Fig.5.2.

This test was applied to the other four test images with reascnable results being obtained. A potential prcblem could be with an image of many small objects or ships, such as ships in a convoy, which would produce regions not close together resulting in the failure of this approach.

C. CCNCIUSIONS

The research work reported in this thesis had as its major objective the extraction of usable target information from infared image data using the recursive image segmentation with hierarchical scope view technique. The five infared images used in the testing were processed sufficiently to yield usable information.

The new procedures of this chapter incorporated into the main program proved tc be inadequate for using its output to make a decision as tc which region is the target. By visual and manual processing this decision was shown tc be possible. Further research will be required to resolve these inadequacies.

There are still many problems that must be solved before this type of procedure can be useful in the field. The

major problem is the processing time. An image that is small with considerable noise present could result in many regions being formed. This could cause the processing time to easily surpass twenty minutes. This is clearly unsatisfactory in any real time situation. A computer engineering research effort to reduce the complexities and excessive nests and loops could be of great benefit.

The region number assignment problem that was dealt with in the new procedures of Chapter IV was not completely remedied. Regions that are unconnected but with the same region symbol assignment occured in all of the tests. The assigning of unique region numbers must be approached from within the main program if this problem is to be resolved. This could possibly be addressed in conjunction with some other research effort utilizing the QUAD SPLIT program.

An initial objective of the research was found to be overly ambitious once the complexities of the programs began to surface. The objective of producing a possible hardware realization of the procedure was therefore not pursued.

APPENDIX A

PROCEDURE BAKREGMAP

```
PROCEDURE BAKREGMAP

    (* This procedure takes the region number
       output of QUAD SPLIT and assigns new
       region numbers from 1 to 64 in a new
       array.  Region number propagation is
       performed, the new region map is output,
       and geometry calculations are made.      *)

  TYPE
    BACKREGIONMAP = RECORD
      BRMAP_PIX : ARRAY [GROW_NO,GCOL_NO] OF GREG_NO
      END;

  VAR
    BREGMAP : ^BACKREGIONMAP;
    I, J, X, Y, Z : INTEGER;

    ENDA :          BOOLEAN;
    FOUND :         BOOLEAN;

  BEGIN
    NEW (BREGMAP);
    WITH BREGMAP^ DO
      FOR I := 1 TO GROW_MAX DO BEGIN
        FOR J := 1 TO GCOL_MAX DO BEGIN
          BRMAP_PIX[I,J] := GREGMAP^.GRMAP_PIX[I,J];
        END;
      END;
      Y := 0;
      ENDA := FALSE;
      Z := 1;
```

```
  FOF X := 1 TO 4096 DO BEGIN
    IF ENDA = FALSE THEN BEGIN
      FOR I := 1 TO 256 DO BEGIN
        FOR J := 1 TO 256 DO BEGIN
          IF BREGMAP^.BRMAP_PIX{I,J} = X THEN BEGIN
            IF Z > 63 THEN Z := 64;
            BREGMAP^.BRMAP_PIX{I,J} := Z;
            FOUND := TRUE;
            Y := Y + 1;
            IF Y >= 65536 THEN ENDA := TRUE;
          END;
        END;
      END;
      IF FOUND = TRUE THEN BEGIN
        Z := Z + 1;
        FOUND := FALSE;
      END;
    END;
  END;
  BAKBND_CK (1);
  OUTBAKMAP;
  GECMETRY;
END;
```

APPENDIX B
PROCEDURE BAKBND_CK

```
PROCEDURE BAKBND_CK (QUADNO : QNODE_NO);
          (* Background boundary check *)

     (*  This procedure segments the image into
         quadrants and then calls the function
         BAKSPLT_CK to check the quadrant
         boundaries for breaks.                  *)

  VAR
    BDR_CLEAN, NS_BDR, FWARD_CK : BOOLEAN;
    BDR_SIZE, BDR_HALF : 32..256;
    BR : GROW_NO;
    BC : GCCL_NO;
    BLEVEL : LEVEL_NO;

  BEGIN
    BLEVEL := QN {QUADNO}^.QLEVEL;
    WRITELN (LISTING, 'QUAD NODE', QUADNO);
    IF QN {QUADNO}^.STATUS = SPLIT THEN BEGIN
      IF QN {QN {QUADNO}^.NW_SON}^.STATUS = SPLIT THEN
        BAKBND_CK (QN {QUADNO}^.NW_SON);
      IF QN {QN {QUADNO}^.NE_SON}^.STATUS = SPLIT THEN
        BAKBND_CK (QN {QUADNO}^.NE_SON);
      IF QN {QN {QUADNO}^.SW_SON}^.STATUS = SPLIT THEN
        BAKBND_CK (QN {QUADNO}^.SW_SON);
      IF QN {QN {QUADNO}^.SE_SON}^.STATUS = SPLIT THEN
        BAKBND_CK (QN {QUADNO}^.SE_SON);
      END;
    NS_BDR := TRUE;
    FWARD_CK := TRUE;
    BR := QN {QUADNO}^.GROW;
```

```
BC := QN {QUADNO}^.GCOL;
BDR_SIZE := 2 ** BLEVEL;
BDR_HALF := BDR_SIZE DIV 2;
BDR_CLEAN := FALSE;
WHILE BDR_CLEAN = FALSE DO BEGIN
  BDR_CLEAN := TRUE;
  WRITELN (LISTING, 'NS FORWARD CHECK');
  IF BAKSPLT_CK (NS_BDR, FWARD_CK) THEN
    BDR_CLEAN := FALSE;
  END;
  WRITELN (LISTING, 'NS REVERSE CHECK');
  IF BAKSPLT_CK (NS_BDR, NOT FWARD_CK) THEN
    BDR_CLEAN := FALSE;
  END;
  WRITELN (LISTING,'EW FORWARD CHECK');
  IF BAKSPLT_CK (NOT NS_BDR, FWARD_CK) THEN
    BDR_CLEAN := FALSE;
  END;
  WRITELN (LISTING,'EW REVERSE CHECK');
  IF BAKSPLT_CK(NOT NS_BDR,NOT FWARD_CK) THEN
    BDR_CLEAN := FALSE;
  END;
  IF BDR_CLEAN = FALSE THEN BEGIN
    IF QN {QN {QUADNO}^.NW_SON}^.STATUS = SPLIT THEN
      BAKBND_CK (QN {QUADNO}^.NW_SON);
    IF QN {QN {QUADNO}^.NE_SON}^.STATUS = SPLIT THEN
      BAKBND_CK (QN {QUADNO}^.NE_SON);
    IF QN {QN {QUADNO}^.SW_SON}^.STATUS = SPLIT THEN
      BAKBND_CK (QN {QUADNO}^.SW_SON);
    IF QN {QN {QUADNO}^.SE_SON}^.STATUS = SPLIT THEN
      BAKBND_CK (QN {QUADNO}^.SE_SON);
    END;
  END;
END;
```

APPENDIX C
FUNCTION BAKSPLT_CK

```
FUNCTION BAKSPLT_CK (NS : BOOLEAN;
                     FORWARD : BOOLEAN) : BOOLEAN;

    (* This function checks each boundary within
       a quadrant for breaks on both sides of the
       boundary.  If this occurs, the adjacent
       regions across the boundary are assigned
       the same region number.  This is referred
       to as region number propagation.          *)

  TYPE
    BRK_TYPE = 1..257;

  VAR
    PREV_TEE, PREV_REG, NOW_REG, NNOW_REG : REG_NO;
    SC, NSC : GCOL_NO;
    BC_LIMIT, BRK, BE_LIMIT : BRK_TYPE;
    SR, NSR : GROW_NO;
    NBRK, PREV_NBRK : -3..255;
    ATEE : BOOLEAN;
    I, J, K : INTEGER;

  BEGIN
    BAKSPLT_CK := FALSE;
    PREV_NBRK := 0;
    PREV_TEE := 0;
    WITH BREGMAP^ DO BEGIN
      IF NS THEN BEGIN
        IF FORWARD THEN BEGIN
          SC := BC + EDR_HALF - 1;
          NSC := BC+ EDR_HALF;
        END;
```

```
    ELSE BEGIN
      SC := BC + EDR_HALF;
      NSC := BC+ EDR_HALF - 1;
    END;
    SR := BR;
  END;
  ELSE BEGIN
    IF FORWARD THEN BEGIN
      SR := BR + EDR_HALF - 1;
      NSR := BR + EDR_HALF;
    END;
    ELSE BEGIN
      SR := BR + EDR_HALF;
      NSR := BR + EDR_HALF - 1;
    END;
    SC := BC;
  END;
PREV_REG := BRMAP_PIX {SR,SC};
NOW_REG := PREV_REG;
IF NS THEN BRK := BR ELSE BRK := BC;
BR_LIMIT := BR + EDR_SIZE;
BC_LIMIT := BC + EDR_SIZE;
WHILE ((NS AND (BRK < BR_LIMIT)) OR ((NOT NS)
        AND (BRK < BC_LIMIT)))
  DO BEGIN
  WHILE (((NS AND (BRK < BR_LIMIT)) OR ((NOT NS) AND
        (BRK < BC_LIMIT))) AND (NOW_REG = PREV_REG))
    DO BEGIN
    PREV_REG := NOW_REG;
    BRK := BRK + 1;
    IF (BRK < 257 ) THEN IF NS THEN
      NOW_REG := BRMAP_PIX {BRK,SC}
      ELSE NOW_REG := BRMAP_PIX {SR,BRK};
    END;
    IF (((BRK - 1) MOD 128) <> 0)
```

```
 AND (((BRK - 1) MOD 64) <> 0)
 AND (((BRK - 1) MOD 32 <> 0))
THEN BEGIN
ATEE := TRUE;
NBRK := BRK -4;
IF NS AND (NBRK < BR THEN NBRK := BR;
IF (NOT NS) AND (NBRK < BC) THEN NBRK := BC;
FOR I := 1 TO 8 DO BEGIN
  IF NS THEN BEGIN
    NPREV_REG := BRMAP_PIX {NBRK,NSC};
    NNOW_REG := BRMAP_PIX {NBRK + 1, NSC};
  END;
  ELSE BEGIN
    NPREV_REG := BRMAP_PIX {NSR,NBRK};
    NNOW_REG := BRMAP_PIX {NSR,NBRK + 1};
  END;
  IF (NPREV_REG <> NNOW_REG) THEN BEGIN
    ATEE := FALSE;
    IF FORWARD THEN BEGIN
      IF (PREV_NBRK < NBRK) THEN BEGIN
        IF NS THEN BEGIN
          FOR K :=BR TO (BR_LIMIT - 1) DO BEGIN
            FOR J:= NSC TO BC_LIMIT-1) DO BEGIN
              IF BRMAP_PIX {K,J} = NPREV_REG
                THEN BEGIN
                BRMAP_PIX {K,J} := PREV_REG;
              END;
              IF BRMAP_PIX{K,J}= NNOW_REG
                THEN BEGIN
                BRMAP_PIX {K,J} := NOW_REG;
              END;
            END;
          END;
        END;
        ELSE BEGIN
```

```
            FOR K := NSR TO (BR_LIMIT-1) DO BEGIN
              FOR J := SC TO (BC_LIMIT-1) DO BEGIN
                IF BRMAP_PIX {K,J} = PREV_REG
                  THEN BEGIN
                  BRMAP_PIX {K,J} := PREV_REG;
                END;
                IF BRMAP_PIX {K,J} = NNOW_REG
                  THEN BEGIN
                  BRMAP_PIX {K,J} := NOW_REG;
                END;
              END;
            END;
          END;
          PREV_NBRK := NBRK;
        END;
      END;
    END;
    ELSE BEGIN
      NBRK := NBRK + 1;
      IF NS AND (NBRK < BR) THEN NBRK := BC;
      IF (NOT NS) AND (NBRK < BC) THEN NBRK:=BC;
      IF NS AND (NBRK > BR_LIMIT - 2) THEN
        NBRK := BR_LIMIT - 2;
      END;
      IF (NOT NS) AND (NBRK > BC_LIMIT - 2) THEN
        NBRK := BC_LIMIT - 2;
      END;
    END;
  END;
  IF ATEE THEN BEGIN
    WRITELN (LISTING,'BREAK AT =',BRK,
             'NEIGHBOR AT', NBRK);
    PREV_TEE := BRK;
  END;
END;
```

```
      PREV_REG := NCW_REG;
    END;
  END;
END;
```

APPENDIX D

PROCEDURE OUTBAKMAP

```
PROCEDURE OUTBAKMAP (* Background Region Map Output *)

    (*  This procedure assigns symbols to the
        region numbers for printer output.  The
        output is formatted into four pieces of
        128 X 128 tc produce the entire output.  *)

  VAR
    RCH : ARRAY [1..64] OF CHAR;
    CHS : PACKED ARRAY [1..64] OF CHAR;
    R, C : INTEGER

  PROCEDURE BLANKLINE (N : INTEGER);

    VAR
      I : INTEGER;

    BEGIN
      FOR I := 1 TC N DO WRITEIN (LISTING);
    END;

  PROCEDURE MAKEDOT;

    BEGIN
      WRITE (LISTING,'   *');
      FOR C := 1 TO 128 DO BEGIN
        IF (C MOD 10) = 0 THEN WRITE (LISTING,'+');
        ELSE WRITE (LISTING,' ');
      END;
      WRITE (LISTING,'*');
      WRITELN (LISTING);
    END;

  BEGIN (* OUTBAKMAP *)
```

```
CHS := '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"()<>
    (* keyboard does not have all the characters
       that VAX-750 has *);
FOR R := 1 TO 64 DO RCH[R] := CHS[R];
BLANKLINE (10);
WRITELN (LISTING,' REVISED GLOBAL REGION MAP');
BLANKLINE(3);
MAKEDOT;
WRITE (LISTING,'    ');
FOR C := 1 TO 64 DO WRITE (LISTING,RCH[C]);
WRITELN (LISTING);
MAKEDOT;
WITH BREGMAP^ DO BEGIN
BLANKLINE (3);
MAKEDOT;
FOR R := 1 TO 128 DO BEGIN;
  WRITE (LISTING,R: 3," ");
  FOR C := 1 TO 128 DO BEGIN
    WRITE (LISTING,RCH[BRMAP_PIX[R,C] MOD 64]);
  END;
  WRITELN (LISTING);
END;
BLANKLINE (3);
MAKEDOT;
FOR R := 1 TO 128 DO BEGIN
  WRITE (LISTING,R:3,' ');
  FOR C := 129 TO 256 DO BEGIN
    WRITE (LISTING,RCH[BRMAP_PIX[R,C] MOD 64]);
  END;
  WRITELN (LISTING);
END;
BLANKLINE (3);
MAKEDOT;
FOR R := 129 TO 256 DO BEGIN
  WRITE (LISTING,R:3,' ');
```

```
    FOR C := 1 TO 128 DO BEGIN
      WRITE (LISTING,RCH{BRMAP_PIX{R,C} MOD 64});
    END;
    WRITELN (LISTING);
  END;
  BLANKLINE (3);
  MAKEDOT;
  FOR R := 129 To 256 DO BEGIN
    WRITE (LISTING,R:3,' ');
    FOR C := 129 TO 256 DO BEGIN
      WRITE (LISTING, RCH{BRMAP_PIX{R,C} MOD 64});
    END;
    WRITELN (LISTING);
  END;
  MAKEDOT;
END;
```

APPENDIX E

PROCEDURE GEOMETRY

```
PROCEDURE GEOMETRY

    (*  This procedure calculates the area,
        perimeter, centroid, size and compactness
        for each of the region symbols used.
        Output format is for printer output.      *)

  TYPE
    XYZ = ARRAY [1..256,1..256] OF INTEGER;
    ZYX = ARRAY [1..64] OF INTEGER;
    YZX = ARRAY [1..64.1..2] OF INTEGER;

  VAR
    IP :        XYZ;
    AREAP :     ZYX;
    CENTER :    YZX;
    PERIM :     ZYX;
    SIZE :      ARRAY [1..64] OF REAL;
    COMPACT :   ARRAY [1..64] OF REAL;
    RCH :       ARRAY [1..64] OF CHAR;
    CHS :       PACKED ARRAY [1..64] OF CHAR;
    J, K, L, R, C : INTEGER;
    PI :        REAL;

  CONST
    MR = 256;
    MC = 256;
    NR = 64;
    NC = 4;

  PROCEDURE AREA1(IP:XYZ;ISX:INTEGER;ISY:INTEGER;
                  IRN:INTEGER;JF:ZYX;NR:INTEGER);FORTRAN;
```

```
PROCEDURE CGRV1(IP:XYZ;ISX:INTEGER;ISY:INTEGER;
                IRN:INTEGER;JF:YZX;NR:INTEGER);FORTRAN;
PROCEDURE PRMT1(IP:XYZ;ISX:INTEGER;ISY:INTEGER;
                IRN:INTEGER;JF:ZYX;NR:INTEGER;
                NC:INTEGER); FORTRAN;

BEGIN
  FOR R := 1 TO 256 DO BEGIN
    FOR C := 1 TO 256 DO BRGIN
      IP{R,C} := BREGMAP^.BRMAP_PIX{R,C};
    END;
  END;
  PI := 3.1415926;
  CHS := '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"()<>
         (* other symbols missing on this keyboard *);
  FOR J := 1 TO 64 DO RCH{J} := CHS{J};
  FOR K := 1 TO NR DO BEGIN
    AREA1(IP,MR,MC,K,AREAP,NR);
    IF AREAP{K} > 0 THEN BEGIN
      CGRV1(IP,MR,MC,K,CENTER,NR);
      PRMT1(IP,MR,MC,K,PERIM,NR,NC);
      SIZE{K} := 2 * AREAP{K} / PERIM{K};
      COMPACT{K} := 4 * PI AREAP{K} / PERIM{K} ** 2;
      WRITELN (LISTING);
      WRITELN (LISTING);
      WRITELN (LISTING);
      WRITE (LISTING,'REGION ',K:4);
      WRITE (LISTING,', SYMBOL ',RCH}K MOD 64}:1);
      WRITELN (LISTING);
      WRITE (LISTING,'AREA ',AREAP{K}:5);
      WRITE (LISTING,'  CENTROID AT ',CENTER{K,1}:3,
             CENTER{K,2}:5);
      WRITELN (LISTING);
      WRITE (LISTING,'PERIMETER ',PERIM{K}:5);
      WRITE (LISTING,',  SIZE ',SIZE{K});
```

```
      WRITE (LISTING,',  COMPACTNESS ',COMPACT[K]);
      WRITELN (LISTING);
    END;
  END;
END;
```

LIST OF REFERENCES

1. Gonzalez,R.C., and Wintz,P., Digital Image Processing, Addison-Wesley, 1977.

2. Lee,C.H., "Recursive Region Splitting at Hierarchical Scope Views", to be published in International Journal of Computer Vision and Image Processing.

3. Shafer,S.A., MOOSE Users Manual, University of Hamburg, 1980.

4. Tamura,H., SPIDER User's Manual, Joint System Development Corp., 1 Dec 1983.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5100	2
2.	Lee,C.H., Code 62Le Naval Postgraduate School Monterey, Califcrnia 93943-5100	3
3.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2

214041

Thesis
B56772
c.1

Bloomquist
Hierarchical image segmentation to infared images.